RL-TR-93-75
Interim Report
May 1993

INTELLIGENT USE OF CFAR ALGORITHMS

Kaman Sciences Corporation

P. Antonik, B. Bowles, G. Capraro, L. Hennington,
A.Koscielny, R. Larson (Kaman)
M. Uner, P. Varshney, D. Weiner (Syracuse University)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
ffiss Air Force Base, New York

RL-TR-93-75 has been reviewed and is approved for publication.

APPROVED: William J. Baldygo, Jr.

WILLIAM J. BALDYGO, JR.
Project Engineer

FOR THE COMMANDER: [illegible]

JAMES W. YOUNGBERG, Lt Col, USAF
Deputy Director
Surveillance and Photonics Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE May 1993	3. REPORT TYPE AND DATES COVERED Interim Jan 92 - Sep 92

4. TITLE AND SUBTITLE INTELLIGENT USE OF CFAR ALGORITHMS	5. FUNDING NUMBERS C - F30602-91-C-0017 PR - 4506 TA - 11 WU - 1C
6. AUTHOR(S) P. Antonik, B. Bowles, G. Caparo, L. Hennington, A. Koscielny, R. Larson (Kaman Sciences Corp), M. Uner, P. Varshney, D. Weiner (Syracuse University)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Kaman Sciences Corporation 258 Genesee Street Utica NY 13502-4627 Syracuse University Dept of Electrical and Computer Engineering Syracuse NY 13244-1240	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (OCTS) 26 Electronic Parkway Griffiss AFB NY 13441-4514	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-93-75

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: William J. Baldygo, Jr./OCTS/(315)330-4049

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The objective of this program is to demonstrate CFAR detection improvement by applying artificial intelligence techniques. The basic concept is one in which a system that dynamically selects the most appropriate CFAR algorithms based upon the characteristics of the interference, out-perform a single, fixed CFAR algorithm. This report describes one prototype Expert System CFAR Processor, its current status, implementation details and capabilities. Also discussed are CFAR algorithm testing and evaluation, and rulebase development.

14. SUBJECT TERMS Radar Signal Processing, CFAR Detection, Expert Systems	15. NUMBER OF PAGES 148
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT U/L

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

1.0 EXECUTIVE OVERVIEW

This is the second interim report for Contract No. F30602-91-C-0017, entitled "Intelligent Use of CFAR Algorithms". The primary objective of this effort is to demonstrate that an expert system CFAR processor, which dynamically selects CFAR algorithms and their parameters based on the environment, can out-perform a fixed, single algorithm processor. The primary emphasis of this report is to document the current status of the Expert System (ES) Constant False Alarm Rate (CFAR) system. This section provides an overview of the ES CFAR program.

Automatic detection schemes are typically employed in operational radar systems. These circuits automatically declare and record target detections from received signals without human interpretation and intervention. Target detections are declared when a signal exceeds a specified threshold level. This threshold is determined by a number of factors such as signal-to-noise (S/N) ratio, probability of detection (P_d), probability of false alarm (P_f), and the statistics of the target and background. For a given S/N, a higher threshold results in a lower P_f but also a lower P_d. Conversely, a lower threshold increases P_d at the expense of more false alarms.

Adaptive threshold techniques are usually employed to control false alarm rates in varying background environments. The most common of these techniques is Constant False Alarm Rate (CFAR) processing. CFAR processors are designed to maintain a constant false alarm rate by adjusting the threshold for a cell under test by estimating the interference in the vicinity of the test cell. A "cell" is a sample in the domains of interest (eg: range, Doppler, angle, polarization). In general the data operated on by the CFAR processor may be pre-filtered to improve detection performance. This pre-filtering may include Doppler filtering, adaptive space-time processing, pre-whitening, and channel equalization.

CFAR algorithms have been studied for many years. The first interim report examined much of that work. A summary is provided in Appendix A. In all, more than 125 CFAR references were considered. Each CFAR algorithm has been designed under a specific set of assumptions, with most CFAR algorithms assuming a Gaussian environment. For example, Cell Averaging (CA) CFAR is designed for a homogeneous, Gaussian, independent and identically distributed (iid) environment. In fact, CA CFAR is optimum under those conditions (provides maximum P_d for given P_f and S/N). Because of its relative ease of implementation and superior performance in thermal noise-limited environments, CA CFAR is one of the most commonly used algorithms. Greatest-Of (GO) CFAR was designed to handle clutter edges and Smallest-Of (SO) CFAR was invented to resolve two closely spaced targets. On the other hand, ordered Statistics (OS) CFAR was designed as a more robust processor. However, the performance of each of these algorithms degrades significantly when the actual conditions vary from the design assumptions. Any single, fixed CFAR is likely to perform inadequately over significant periods of time for a wide area surveillance sensor.

The objective of this program is to demonstrate CFAR performance improvement by applying artificial intelligence techniques. The basic concept is that a system that dynamically selects CFAR algorithms and controls CFAR parameters based on the environment should out-perform a single, fixed CFAR system. The ES CFAR system is expected to have a high payoff in:

- Dynamic environments (eg: moving platforms)
- Non-Gaussian backgrounds
- Clutter edges (eg: land/sea interfaces)

A fully developed ES CFAR system would utilize a variety of "knowledge sources" to ascertain characteristics of the radar data. Geographical maps combined with radar location and pointing data may provide information regarding clutter edges. Statistical distribution identification algorithms may provide information regarding the statistical nature of the data. A tracker may provide important information regarding multiple targets. Also, the user may supply other valuable inputs to the system.

"Rules" of the ES CFAR system translate the above information into action. Based on the observed state of the environment the rules determine which CFAR algorithm or algorithms are executed. They also dynamically determine the appropriate CFAR parameter values (eg: window size, order number). The rules may also infer new information from the known information.

The majority of this report describes the status of the ES CFAR system development. Figure 1-1 shows a block diagram of the prototype/demonstration system. The top portion of the diagram is labeled "Baseline CFAR" and represents conventional CFAR processing. In the baseline processor the radar data passes directly to the CFAR algorithm. In general, this radar data is pre-filtered which may include Doppler filtering or adaptive filtering. The detections resulting from the baseline CFAR are then passed to the output displays such as the PPI.

The lower portion of Figure 1-1 shows the ES CFAR processor. As in the baseline processor, the radar data is passed to the CFAR algorithms. However, in the ES path the data is also processed by a Statistical Distribution Identifier to extract statistical features from the data. This information may be combined with user inputs, geographical data, and radar location and pointing information and used to classify the environment by statistical distribution, homogeneity, and terrain features. Target information is supplied both by the user and from a tracker feedback loop. Tracker feedback may indicate, for example, the presence of multiple targets. Clutter and target information is then sorted and weighted to select the CFAR algorithm or algorithms to be executed, along with their parameters. The outputs of the selected CFAR algorithms are then combined in the Weighting and Fusing Circuit. In general, the actual false alarm rate will differ from the design value since the environment is unlikely to satisfy all of the CFAR design assumptions. The False Alarm Control block attempts to reconcile these discrepancies. Outputs from that circuit are finally passed to the Performance Monitoring and display functions, as well as the tracker loop which updates known target information.

At the end of the current contract the ES CFAR system will:

- Process measured airborne radar data
- Demonstrate improved CFAR processor performance
- Demonstrate the application of AI techniques to radar signal processing
- Provide a testbed for CFAR algorithm development.

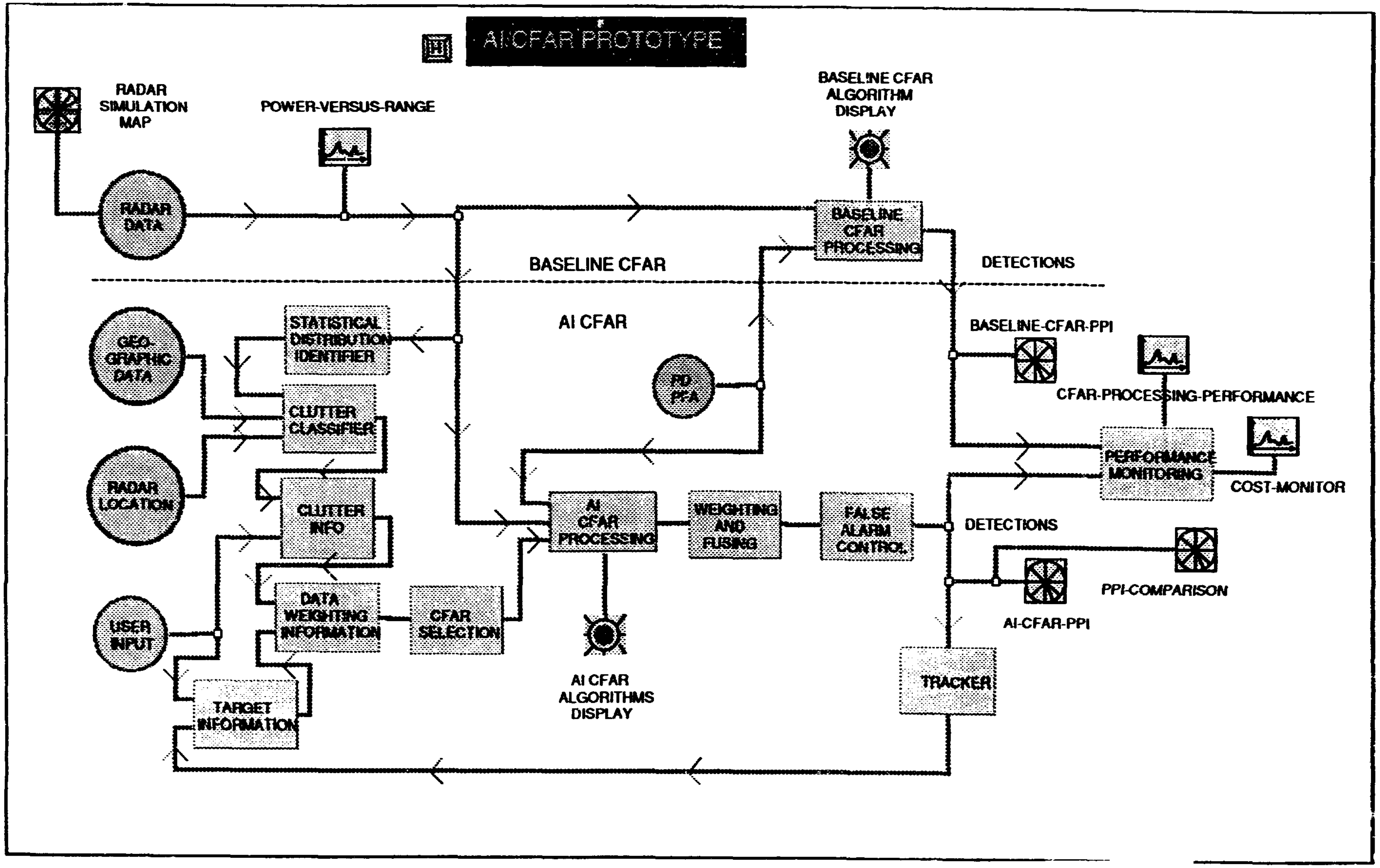


Figure 1-1: Schematic Diagram of the ES CFAR expert system. The ES CFAR system dynamically selects CFAR algorithms and parameters based on the environment.

Upon successful demonstration of the ES CFAR concept, longer range plans include the implementation of an ES CFAR system into an operational radar for a real-time proof-of-concept validation experiment. Several platforms are being considered for this experiment, with emphasis being placed on airborne wide area surveillance radars. These platforms typically have sufficient field-of-views to ensure diverse background environments. Further, a moving airborne platform ensures a dynamic environment to challenge the CFAR processor.

In the future, substantial performance improvements will likely not be the result of higher power-aperture products. Larger antennas and higher peak powers will be more difficult to achieve, especially for platforms limited in size, weight, and power. Substantial performance gains will more likely be the result of advanced processing techniques. These techniques will extract more information from received signals using the available power-aperture product. One of the advanced processing thrusts to further that goal is to transition recent artificial intelligence advances into the radar signal processor. This contract has focused on one portion of the radar processing chain: the CFAR processor. This is only a first step in the AI technology transfer and one step towards sizable performance improvements through advanced signal processing.

2.0 KBS DEVELOPMENT

The first interim report for this program discussed an initial system design for the AI CFAR system. The proposed system design has since evolved with time. While the overall design philosophy remains intact there have been some modifications to the system structure. The processes previously referred to as the Local Contrast Filter and the Clutter Classifier no longer exist as separate processors. The functions provided by the Local Contrast Filter have been combined with new functions and placed in the pre-processor. The pre-processor operates on the radar data prior to the CFAR processor and is designed to filter the radar data and to separate it into relatively homogeneous regions. In the original design the Clutter Classifier determined the clutter type from the available information sources. A set of rules then determined which CFAR algorithms were executed based on the clutter type. In the current system design, however, the intermediate step of determining clutter type is eliminated and the rules determine directly which CFAR algorithms are executed based on the known and derived information. The derived information includes statistical estimates. Computer code to estimate these statistics has been partially implemented and tested and is discussed further in Section 2.3.

Changes have also been made in the implementation. It was recognized early on that computationally intensive parts of the system would only slow down the rule evaluation process performed by Gensym's G2 and that it made more sense to implement these functions in remote procedures. Remote procedures refer to those calculations which are performed outside of G2, which is the Expert System development software. All AI components of the system remain in G2. The user interacts exclusively with G2 which passes commands and controls to the remote procedures. Functions such as clutter simulation, radar simulation, and actual CFAR processing are implemented primarily in the 'C' programming language. Every attempt has been made to create a simpler and faster system by reducing the data transferred between these processors.

This report discusses the "System" as it is planned to be at the end of this effort. However, the report also makes clear which parts of the system are actually complete and which parts are either in development or will be developed in the future. Currently, the Baseline and the ES CFAR paths through the system have been completed, exclusive of the rules. That is, the framework is in place to permit

CFAR rule evaluation. As reported earlier, the domain data gathering process has been completed and testing of the rules has begun. However, rule testing has not reached the point where significant progress can be reported. Therefore, this report discusses primarily those parts of the system which have been implemented and tested. Parts of the system which reside in G2 will be described in G2 terminology. Those terms may be foreign to those who have not been exposed to G2. For this reason, background information on Expert System development and G2 is included which will help the reader have a better feel for how the ES CFAR Knowledge Based System (KBS) works.

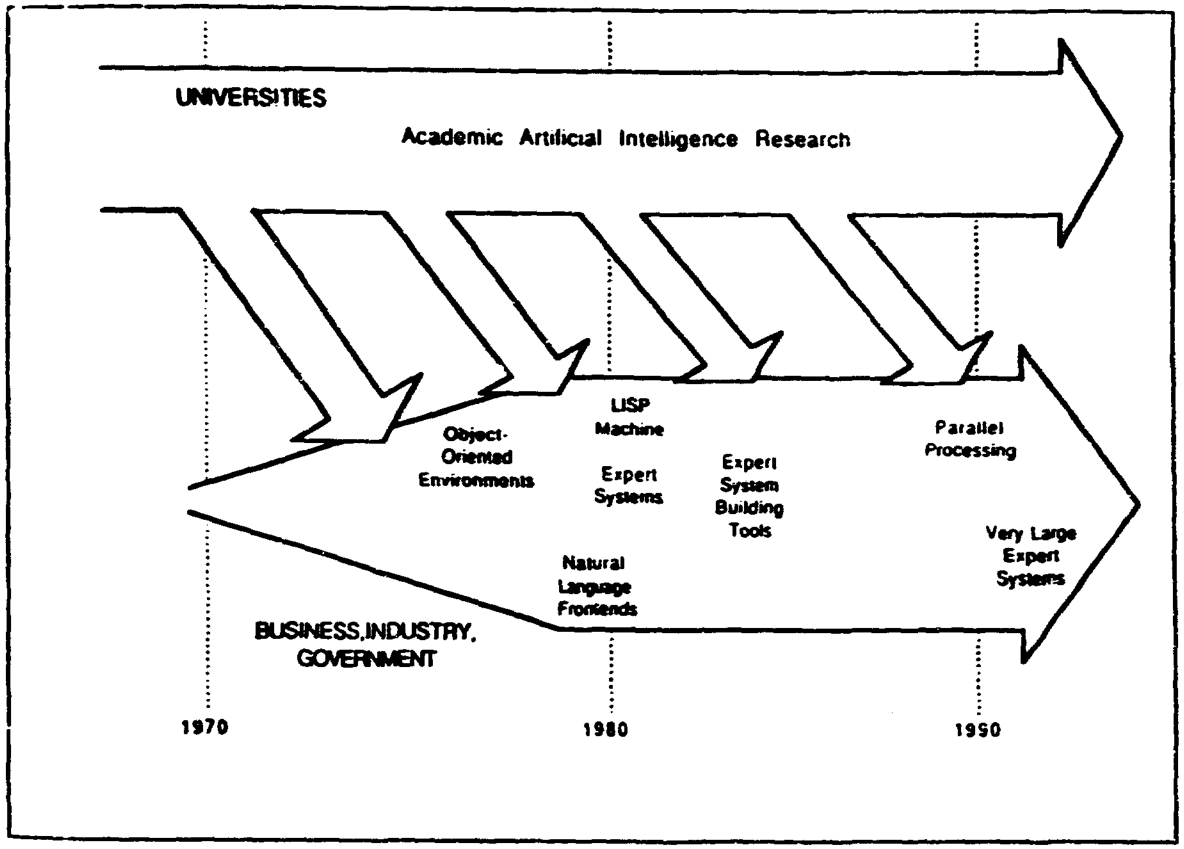
2.1 Expert System Development

The evolution of computer languages is illustrated in Figure 2-1. Assembly language was dominant between World War II and 1960. In the late 1950s, higher order languages (HOLs) began to emerge. FORTRAN was one of the first HOLs and was developed as a scientific programming language.

In the early 1960s programming languages began to take two paths: symbolic languages and algorithmic languages. Symbolic languages are oriented more towards logic and list (eg: text) processing than numerical processing. One of the first and most common symbolic languages was LISP. A number of algorithmic languages were also developed between 1960 and 1985 including FORTRAN, PL1, PASCAL, C and ADA.

Computer languages began to branch out again in the early 1980s. Symbolic languages began to expand into PROLOG, LISP, and Object-Oriented approaches. Meanwhile, algorithmic languages began to incorporate symbolic language features. For example, C++ is an extension of C, based on object-oriented programming. Also during this time, so-called "Fourth Generation" languages began to emerge. Fourth Generation languages again raised the level of abstraction and are commonly associated with database management systems. By 1985 the programmer had a host of languages available in a number of varied approaches.

The emergence of Expert Systems and object-oriented environments led to the development of Expert System Building Tools, such as Gensym's G2. This is illustrated in Figure 2-2. Also note in Figure 2-2 that academic research into


UNIVERSITIES
Academic Artificial Intelligence Research
Object-Oriented Environments
LISP Machine
Expert Systems
Expert System Building Tools
Parallel Processing
Natural Language Frontends
Very Large Expert Systems
BUSINESS,INDUSTRY, GOVERNMENT
1970
1980
1950

P. Harmon, R. Maus, & W. Morressey,

<u>Expert Systems Tools & Applications</u>

Figure 2-1: The evolution of computer languages. Unlike in 1960, the programmer today has available a number of languages in a wide variety of approaches.

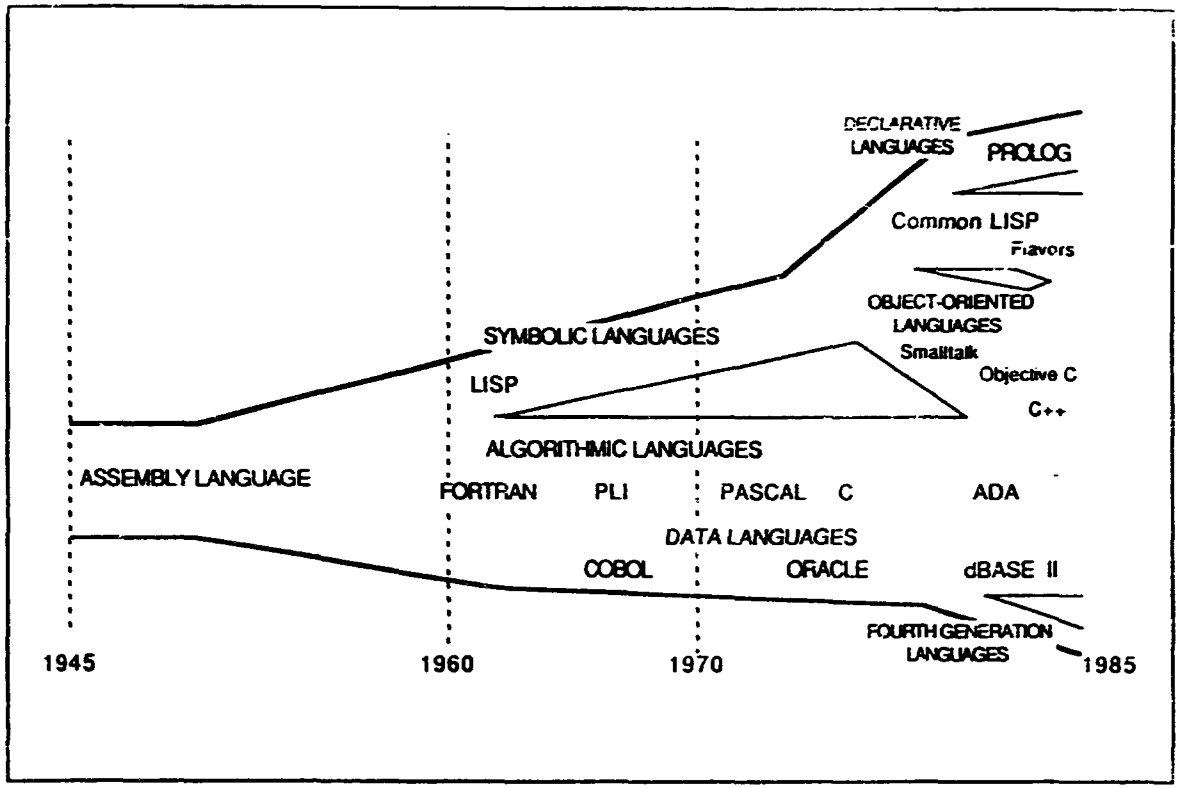


***Figure 3.2** The evolution of computer languages.*

P. Harmon, R. Maus, & W. Morressey,
<u>Expert Systems Tools & Applications</u>

Figure 2-2: Recent AI advances. The development of expert systems and object-oriented environments led to the development of expert system building tools, such as Gensym's G2.

Artificial Intelligence (AI) is leading and feeding practical AI developments in business, industry, and government. Today, some of the leading AI activities are in the areas of distributed processing and Very Large Expert Systems, which includes developments in "Blackboard" technologies.

Is the Object-Oriented approach "better" than other approaches? Not necessarily. However, the object-oriented approach is appealing for a number of problems, including the ES CFAR system. A key point being made here is that, unlike 1960, the programmer today has a variety of approaches available. The applications programmer (eg: signal processing algorithm developer) needs to be aware of these advances in AI in order to fully exploit the new computer technologies.

One goal of this report is to describe the specific expert system tool being used to build the ES CFAR application. This discussion begins with a general description of how expert systems work, describing their components and how the components interact.

2.1.1 Expert Systems

An expert system can be described as a computer based system that uses reasoning techniques to automatically search, analyze, and exploit knowledge that has been appropriately represented within the computer. With conventional computing techniques, reasoning based search and analysis tasks must be performed in the minds of the human expert. The success of expert systems depends on their ability 1.) to store or represent as a computer knowledge base the domain knowledge, associated with "human expertise", in a manner that enables logical inference ("the knowledge representation problem") and 2.) to search, within reasonable time constraints, through this knowledge base for solutions to interactively posed "domain" problems ("the search problem"). While expert system designs can vary, they all tend to have a knowledge base, an inference engine, and an interactive user interface.

Knowledge Base

The knowledge base may contain fundamental generic knowledge, as well as generic and specific knowledge from the human expert. Fundamental knowledge for example might be rules for sorting data, independent of the domain, into alphabetic or increasing numeric order. Within a domain, the experts are encouraged to state their knowledge as generically as possible. This tends to increase its applicability and also reduce the amount of knowledge to be searched. For example, the "inverse square law" of wave propagation might be repeatedly encoded, as rules, at the physical phenomenology level, i.e. for sound, light, and radio waves. A better approach is to have a generic representation that can be applied with any wave propagation phenomenology.

Nevertheless, many rules will have to deal with more specific cases in the domain. For example, a rule that inferred that all interstate highways had a maximum speed limit of 55 miles per hour would be too generic. Such a rule, would have to be specialized to look at individual states and at proximity to major populated areas within some states. Most expert systems rely primarily on rules as their knowledge representation formalism. Rules have the form "If A then B". This means if A is true then infer that B is true. Another rule might be "If B then C". Once the truth of A is established the inference engine will search through two rules to conclude that C is true. Of course, this example is overly trivial. The complexity begins when the left and right sides of the rules have multiple parts, such as, "If x and y then d and e." or "If x or y then d and e."

Facts are also a major ingredient of the knowledge base. A fact can be thought of as a rule of the form "If true then pi = 3.14159". The left hand side of such rules are always trivially true, i.e. "true is true" is a tautology. Therefore, the left side of the rule can be deleted and the right hand side simply asserted as a fact, i.e. pi = 3.14159. The inference engine, however, processes facts as if they were rules whose left hand conditional has already been satisfied. Facts can also be generic or specific. The "pi" example is generic, but other facts might be more specific such as "The Gooseneck-Radar-Frequency = 1.5 gigahertz." Facts should be items that are not conditionally dependent on anything else within the context of the domain problem. While a Gooseneck radar may be modified for a different frequency, within the context of the current problem it will not be so modified Obviously, some facts are more secure than others. The value of "pi" is a safer fact, then the assertion of a system's radio frequency.

Inference Engine

As already implied, the inference engine is an algorithm that concludes new truths from known truths. The process is deductive inference in that the results are always correct. Inductive and abductive inference by contrast can only suggest solutions that may be true, i.e. hypotheses.

When the inference engine proceeds, as in the above example (from the truth of A, to the truth of B, and finally to the truth of C), this is called "forward chaining". In this case, each rule is applied from left to right. The rules can also be applied in the reverse direction which is called "backward chaining".

Most expert systems use backward chaining because they are goal driven, i.e. given that the desired solution is C, how can the system find a path to C. In this case, the rule "if B then C" would be applied first with the conclusion that a way must be found to show that B is true. Next, "if A then B" will be applied because the right hand side matches with the B goal. Now, given that A is a "fact", the search is complete. However, as mentioned above, the inference engine actually treats the fact "A" as a rule, i.e. "if true then A." Thus, the inference engine actually backward chains one more step, matching the left hand side of "if A then B" with the right hand side of "if true then A." The inference engine therefore responds with "true", meaning that the goal state C is true.

For most problems, backward chaining seems to result in more efficient, meaning faster, searches by the inference engine. The desirability of forward versus backward chaining depends on the forward and backward branching factors for the rules. This relates to how many "and/or" components exist in the left and right side of the average rule. The trick is to use the direction that generates the smaller expansion of possible search paths. While most expert systems perform better with backward chaining, expert systems for manufacturing control and other planning problems, tend to perform better with forward chaining. This is because they are constrained by their starting conditions but can accept multiple solutions or end conditions. A backward chaining scheme corresponds to a constrained goal with multiple acceptable starting conditions.

Interactive User Interface

An expert system normally has two types of interactive users: the domain expert and a non-expert user of the domain knowledge. Through the interactive interface, which is often a graphical user interface with easy-to-use "pull down" menus, the expert can insert, critique, and modify the domain knowledge of the system. The non-expert user can insert problems and receive solutions, along with a rationale or logical explanation. The logical explanations are actually the knowledge inserted by the domain expert. In this way, the user can go to the system for consultative support rather than going to the expert. In this sense, the system becomes a productivity multiplier for the human expert. The explanation capability is a key aspect of expert systems technology. The system is not only supposed to provide a solution, it is supposed to convince the user that the solution is correct.

Expert System Development Shell

Since the shell provides all of the deductive reasoning algorithms for the inference engine, the system developer can concentrate or focus on the specification of declarative domain knowledge for the knowledge base, rather than on the development of search strategies.

Key to the development of an expert system is the description of the knowledge an expert uses to solve a problem. There are a number of ways to represent this knowledge in a knowledge base. While rules are the easiest to understand from a deductive inference viewpoint, knowledge can also be represented as frames, semantic networks, scripts, relations, or even procedures. Various expert system environments support many of these alternatives, in addition to rules. Some of these formalisms, however, while adding run-time efficiency and even ease of knowledge capture are not amenable to automated inference. In particular, a procedural representation can not be checked by the inference engine for internal consistency--an illogical programming step ("bug") will normally not be detected by the inference engine as it might be in a rule based formalism. Nevertheless, procedures are pragmatically necessary.

The extension of knowledge representation techniques, beyond the rule formalism, is nicely facilitated by object oriented programming (OOP) methods. Object oriented systems provide a simple, natural way of capturing relations which can be used to represent rules, semantic nets, and other formalisms. In object oriented programming, the programmer defines classes of objects for an application. Each class definition includes a programmer-provided list of attributes. Objects can then be dynamically created as instances of the class. Each instantiated object can have different values assigned to the attributes. The classes can and should exist within a class hierarchy to facilitate inheritance. All attributes should be defined at the highest possible class level in the hierarchy. This means they should be as generic as possible. Each lower level class inherits the attributes of its superior parent , but it can also contain its own, more specific, attributes. While classes and class hierarchy structures are defined solely by the developer, as a rule of thumb, most classes are defined in ways that strongly parallel their existence in the real world. Thus, the program successfully models the real world relationships.

Using the object oriented approach gives the developer the ability to structure and easily define a large number of facts and relationships without having to incorporate them as rules. Rules can be reserved for those cases where they are the more natural representational form, i.e. when the heuristic knowledge in the domain has a natural "if.... then ..." rule format. Such heuristics, while often based on incomplete evidence, can easily capture a human expert's "rule-of-thumb" intuition about the domain knowledge.

Inference and Control

Expert system programming involves finding the appropriate description of domain knowledge so that it can be represented declaratively, rather than procedurally within the system. This means that minimal programming is required to utilize the knowledge; the inference engine provides the programmatic logic. The control and search strategies that the inference engines employ are pre-developed and are not typically governed by the developer. In this way, the same inferencing techniques can be applied for the control of many kinds of knowledge bases and applications.

Inference engines are typically described by the types of inference and control strategies that they employ. One or more inference engine control strategies may be available in a particular expert system environment. The most common techniques are forward and backward chaining, described above, and breadth first versus depth first search.

The latter option addresses how the system will cope with branching factors at each level of rule application in the inference cycle. Breadth first means that all rule options are generated at each level before proceeding to the next level. Depth first means that only one rule is applied at each level before proceeding to the next level. Depth first will on average find a solution in half the search time of breadth first search. However, depth first must know a priori how many levels exist. If it stops one level short of a solution it won't find it. When depth first reaches the lowest level without finding a solution, it backs up one level and tries the next matching, but not yet tried, rule. It then proceeds down again in a depth first mode. Eventually, the system, by backing up and going forward again, will exhaust the available rules at all levels. i.e. That is, if it backs up and there are no more untried rules at that level, it backs up again, eventually reaching the top before proceeding down again. While breadth first takes twice as long on average, it does not have to know in advance how many layers to try. It exhausts all rule applications at each layer before proceeding to the next layer.

Other control strategies are also available such as heuristic search where meta-rules are used to prune unlikely, though legal inferences, from the exponentially expanding search tree. Rules can also be grouped into hierarchically related sets so that at different levels in the tree, the number of rules that have to be linearly checked for a match is reduced. Hashing and other specialized pattern matching techniques can be used to increase the efficiency of the search process. Some techniques are more closely associated with formal logic, specifically first order logic or predicate calculus. This set of techniques is now generally known as logic programming. The inference step for this formalism is known as resolution and the binding from rule to rule in this inference process is controlled by unification. A popular system for logic programming is Prolog whose resolution based inference is equivalent to a backward chaining, depth first search.

In summary, inference systems can be thought of as deductive, inductive, or abductive:

- Deductive systems apply rules to derive absolute, but non-explicit, hidden truths from explicitly known truths.

- Inductive systems try to find rules by searching for correlations between factual events over a large sample size. This process, unlike deduction, is not fail safe. Correlating sunlight with growth can correctly lead to the rule that "if the sun shines then the plants grow." However, other, seemingly plausible correlations, can lead to illogical rules such as "If the sun passes over head once a day then it circles the earth once a day."

- Abduction generally applies known rules in a backward chaining hypothetical sense. It can not prove the truth of an earlier event, but only the possibility that something might have occurred which caused the later event to be true.

Expert systems have been built that address all three of these inference types. Induction and abduction usually require a second level of abstraction in the inference engine. Most expert systems, and virtually all applied expert systems, are restricted to deductive logic. While the programmer and the domain expert may practice inductive inference to find the rules, this is normally a human rather than a machine inference process. The machine restricts itself to the deductive application of the given rules. It may, however, challenge the validity of the human inductive mechanism's results when two rules are found to be mutually inconsistent, but applied systems do not normally infer their own rules. Inductively inferring rules is actually a form of machine learning; it is currently an active and exciting research area.

In addition to the variability of inference mechanisms, knowledge representation techniques can also vary from a strict compliance with the rule-based formalism to the incorporation of frames, scripts, semantic networks, and other relational strategies. Hybrid systems are rapidly evolving that allow the advantages of a rule based deductive inference formalism to be combined with other advanced representational strategies as well as with conventional programming techniques.

Object oriented environments provided the necessary structure for such hybrid systems. In these systems, a core module can provide the rule-based inferential control, while allowing the lower level processes, such as conventional digital signal processing, to remain as sub-routines in conventional procedure based languages. This facilitated by the fact that procedures and rules can all be thought of as objects and thus easily integrated by the object oriented environment.

2.1.2 Gensym G2

Gensym's G2 is an expert system tool which is being used to prototype and develop the ES CFAR expert system. It is a complex system which uses the power of object-oriented programming with a hybrid inference engine and the use of both rules and procedures in order to offer one of the most flexible environments possible for systems-development. Application development is done in a graphical, object-oriented environment with support tools which provide for both rapid prototyping and full scale system development. Gensym's G2 was selected for prototype development because of its professional-level capabilities, graphical user interface and development environment, and its availability of support.

The first step in developing an application with G2 is defining the class of each object in the application's knowledge domain. In the ES CFAR application, the key class definitions include radars, clutter, targets, CFAR-processes, and detections. Along with defining the object classes and their attributes, one must also define the class hierarchy.

The class hierarchy identifies how certain objects are related to others and how objects inherit attributes from their superior classes. Consider, for example, the class of objects named clutter. The class clutter (within the ES CFAR application) has an attribute which defines whether the clutter is Weibull, exponential, log-normal, etc. Each of these specific subclasses of clutter have attributes that are descriptive of their "statistical distribution" such as mean and variance. Each subclass also inherits the attributes of its superior class. Another example from the ES CFAR application which describes how class definitions identify relationships between objects can be seen in how the class of radar objects is defined. A radar has subsystems and characteristics defined as attributes of the radar: antenna, receiver, transmitter, beam, and direction. Each of these subsystems is itself a unique class of objects with unique attributes that describe each specific subsystem. By defining each subsystem class as an attribute of the class "radar", each instance of a radar includes an instance of each the subsystems.

Once the developer has defined the class hierarchy and all classes, a model of the application is created. The model includes all relationships, objects, rules, and procedures that describe the application. The developer organizes this knowledge in the knowledge base by placing related items on a single workspace. Workspaces are the places in the knowledge base where the items which make up the knowledge base are located. The workspaces are also structured in a hierarchy of subworkspaces. One way the developer may organize the items in the knowledge base is to have a superior workspace as a menu with selections such as rules, object definitions, procedures, user-menu-choices, etc. Each of the selection objects on the menu have a subworkspace, where a collection of like objects are collocated.

The workspaces, objects, rules, procedures, etc., make up the knowledge base. All this knowledge describes how the attributes of the objects are related and how they can receive their values.

G2 offers sources which determine how attributes can uniquely receive their values. Attributes can receive their values from the inference engine, the G2 simulator, through "real-world" sensors, and through the G2 Standard Interface (GSI). The inference engine uses its inferencing mechanism to determine the values of attributes. The G2 simulator produces values through the use of simulation formulas. When attributes receive true values through "real-world" sensors, the GSI is used to connect the G2 variables with external sensors. The GSI is also used to connect G2 with external data sources such as with remote processes. In the ES CFAR application, the remote procedure call (RPC) is used frequently to obtain values for variables (through the GSI) from remote C-programs. In the ES CFAR application, some of the key processes are numerically intense and need to be executed in a remote environment. For instance, the radar data is processed by a CFAR algorithm in order to report detections. In this case, G2 determines which CFAR process should be used to process the data and sends this information through an RPC to the CFAR process which is coded in C language. The CFAR process or outputs detections, which are sent back to G2 through another RPC.

In the ES CFAR application, as well as in a wide variety of other applications, the best solution for a problem results from a combination of symbolic and procedural programming. Knowledge engineers must analyze the problems and decide how to distribute them between procedural code (either in G2 or remote procedures) and expert system techniques. In the ES CFAR system the use of remote procedures in the form of C code results in a faster system as well as the ability to use statistical routines not provided in the expert system development shell (G2).

2.2 KBS Overview

A variety of diagrams have been developed to design and illustrate the ES CFAR system since both procedural and symbolic aspects need to be represented. Figure 1-1 showed an overall schematic of the current ES CFAR system that emphasizes data flow and data displays. Beginning at the upper left of Figure 1-1, radar data flows to the baseline CFAR and ES CFAR processes. The baseline CFAR path represents conventional CFAR processing and is indicated by everything above the horizontal dashed line. The radar data may be either simulated or measured. A display of received power versus range is available by "clicking" on the POWER-VERSUS-RANGE icon. The baseline process exercises a user selected CFAR algorithm on the same data as the ES CFAR process for comparison purposes. The output of the baseline process is a list of detections, which can be displayed in PPI format ("click" on BASELINE-CFAR-PPI) and which flows to the performance monitoring process for calculation of actual P_d and P_f. The CFAR-PROCESSING-PERFORMANCE icon brings up a comparison display of time histories of P_f and P_d for the baseline and ES CFAR processes.

The radar data also flows into the ES CFAR process. Prior to actual ES CFAR processing, the radar data flows to a series of processing routines (the pre-processor) that find discretes, classify the radial of data into relatively homogeneous clutter regions, and integrate additional sources of information such as geographic data, radar position information, user input and knowledge about the target. This information is used to select the CFAR algorithm(s) to be used for the various clutter regions. A list of clutter regions and the associated CFAR algorithms then flows to the ES CFAR process. The ES CFAR process exercises the CFAR algorithms on the radial of data and produces a list of detections. Currently, Cell Averaging, Greatest-Of, Trimmed Mean, and Ordered Statistics CFAR algorithms are being implemented. The WEIGHTING AND FUSING and FALSE ALARM CONTROL processes will then operate on this list of detections to produce the final list of ES CFAR detections. These two processes have not yet been implemented. The list of detections flows to the PERFORMANCE MONITORING circuit so the actual P_f and P_d for the ES CFAR process can be computed and displayed. As with the baseline process, a PPI display of detections is available from the AI-CFAR-PPI icon and the time history of P_f and P_d is available from the CFAR-PERFORMANCE-MONITORING icon. The TRACKER process will operate on the list of detections and maintain a target track, when it is fully implemented.

While the basic data flow through the ES CFAR system can be seen in Figure 1-1, another diagram is needed to show where these processes are implemented and how the data is transferred between them. Figure 2-3 presents this kind of information. As noted earlier, several software tools are being used in the ES CFAR system. The user interface, process control, and inferencing are performed by G2 and include the following items in Figure 2-3:

- Overall Process Control
- Radar Parameter User Interface
- CFAR Parameter User Interface
- Data Weighting
- CFAR Selection
- Detection - Weighting-and-Fusing
- False-Alarm Control
- Performance Monitor
- P_d - P_f - Statistics

Computationally intensive processes are implemented using remote procedures, called through GSI. These remote procedures are being developed in C and include the following functions in Figure 2-3:

- Radar Data Generator
- Statistical Distribution Identifier
- Clutter Classifier
- Baseline CFAR Process

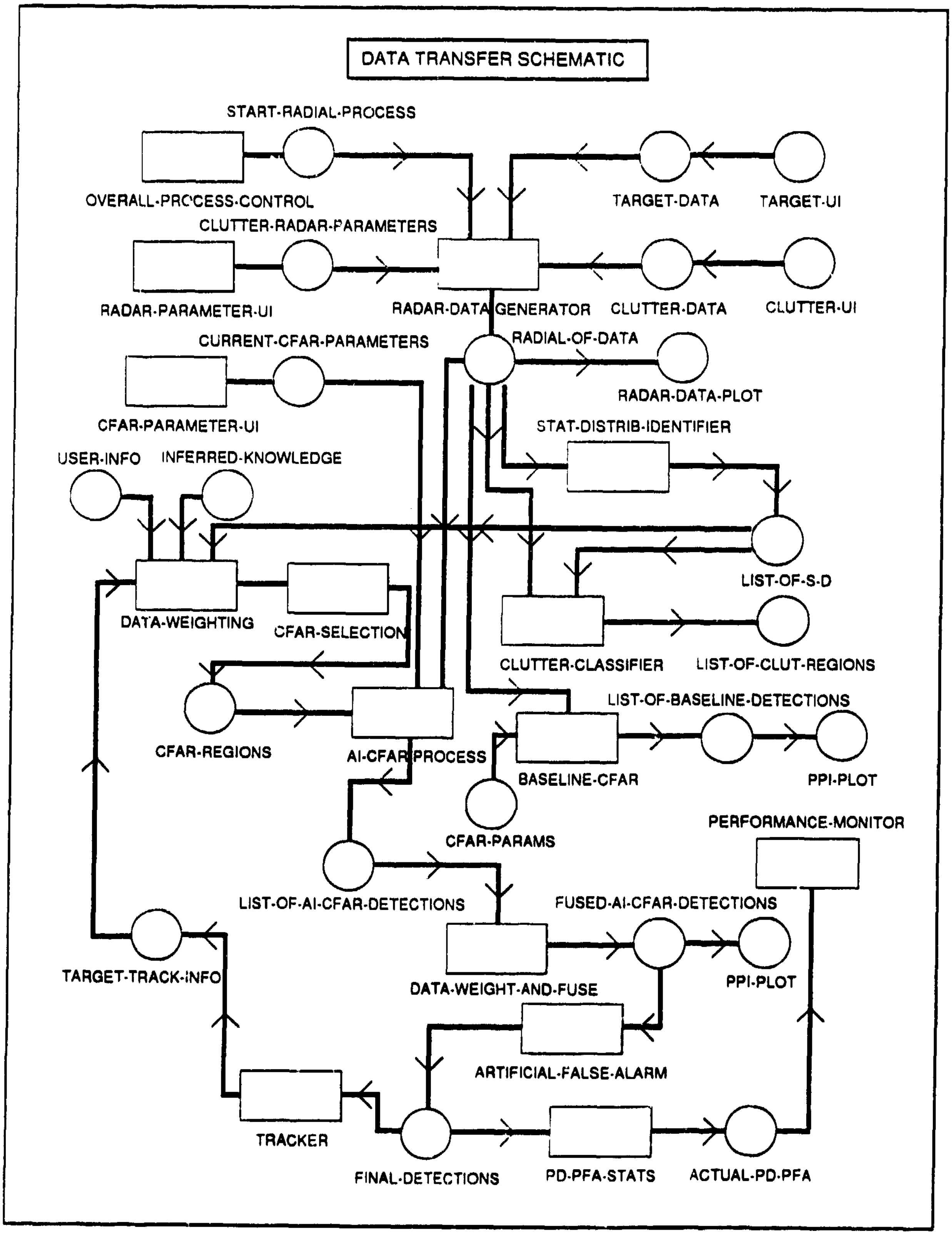


Figure 2-3: Data Transfer Schematic. The diagram shows where various processes are implemented and how data is transferred.

- ES CFAR Process
- Tracker

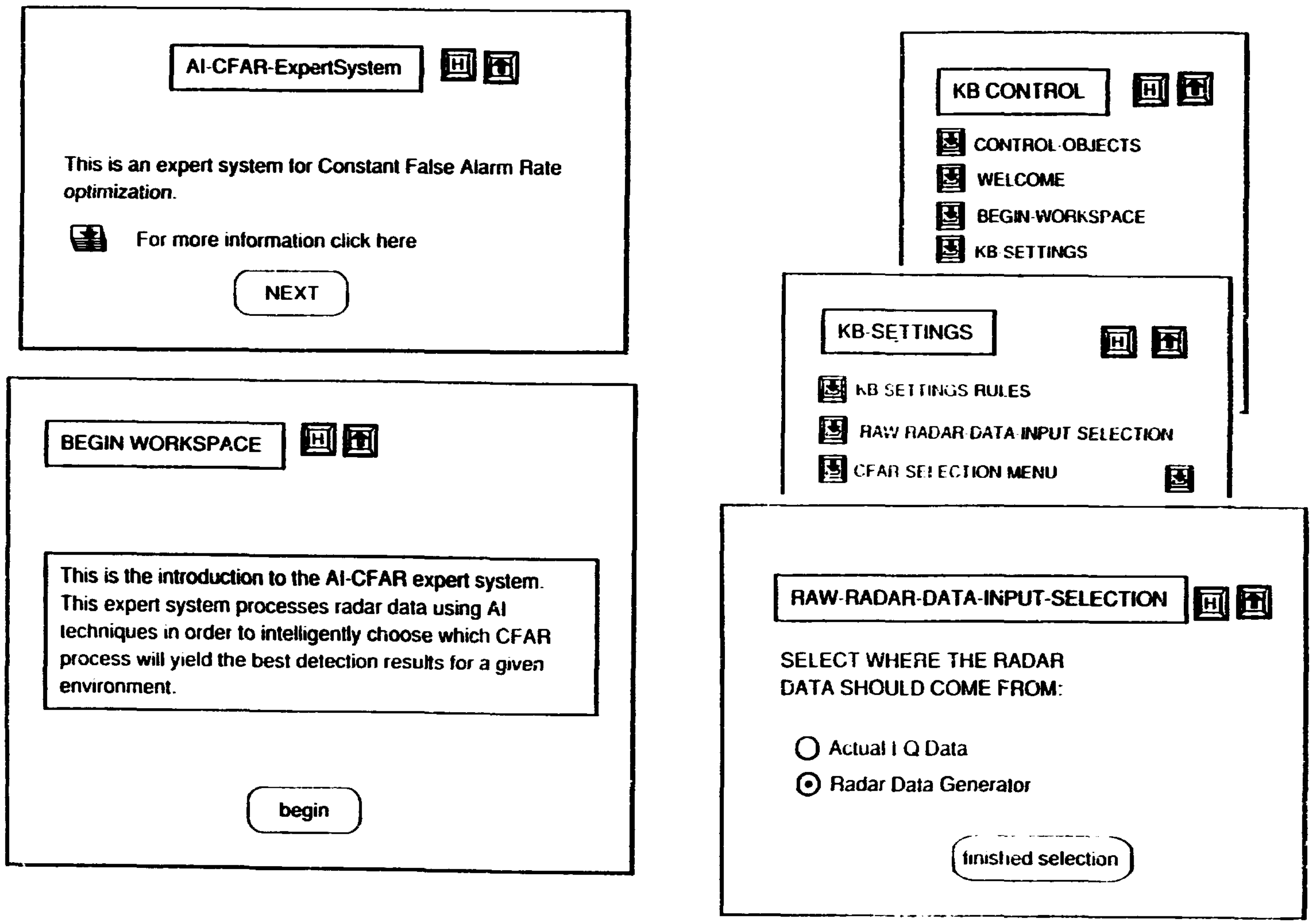
Processes that need a quality graphical display capability are implemented using a commercial graphics package called PV-WAVE. These processes include:

- Target User Interface
- Clutter Map User Interface
- Radar Data Plot
- PPI Plot

The remaining items include various data transfer items needed for inter-process communication.

Examples of G2 subworkspaces are shown in Figures 2-4 through 2-7. The user begins the simulation from the G2 main menu by selecting "Get Workspace" and then selecting the "Welcome" workspace, shown in the upper left panel of Figure 2-4. Desired subworkspaces then appear automatically or are user selectable as the user sets up the simulation. The source for the radar can be selected (i.e., simulated or real), although the "actual I-Q data" selection has not yet been implemented. Figure 2-5 shows the subworkspaces and menus for setting up the radar. Predefined radars can be selected and radar parameters can be modified through the radar subsystems tables. Next the CFAR selection menu appears as shown in Figure 2-6. One CFAR algorithm can be chosen for the baseline CFAR. Any or all of the CFAR's can be chosen as candidates for the ES CFAR selection rules. Currently, simple rules only for testing purposes are being used (e.g., always choose OS). Next the subworkspace of "Simulation Map" appears as shown in Figure 2-7. The user can place targets on the map and change clutter attributes by clicking on the CFAR-TESTING-CONTROL button. The actual simulation begins when the "Simulation On" button is selected.

G2 begins the radar simulation process by calling the appropriate remote procedures. As the user defines the simulation via menus in G2, data is sent to the remote procedures and is stored in global variables. For example, when the radar has been defined by selecting a predefined radar and perhaps changing some of its parameters (as shown in Figure 2-5), the information is passed to the remote



This is an expert system for Constant False Alarm Rate *optimization.*

This is the introduction to the AI-CFAR expert system. This expert system processes radar data using AI techniques in order to intelligently choose which CFAR process will yield the best detection results for a given environment.

SELECT WHERE THE RADAR DATA SHOULD COME FROM:

Actual I Q Data
[✓ Radar Data Generator]

Figure 2-4: G2 user interface subworkspaces. The welcome workspace, upper left, can be displayed from the main menu. When the user selects "NEXT", the introduction subworkspace at lower left appears. At lower right, the user selects the source of radar data as either measured or simulated.

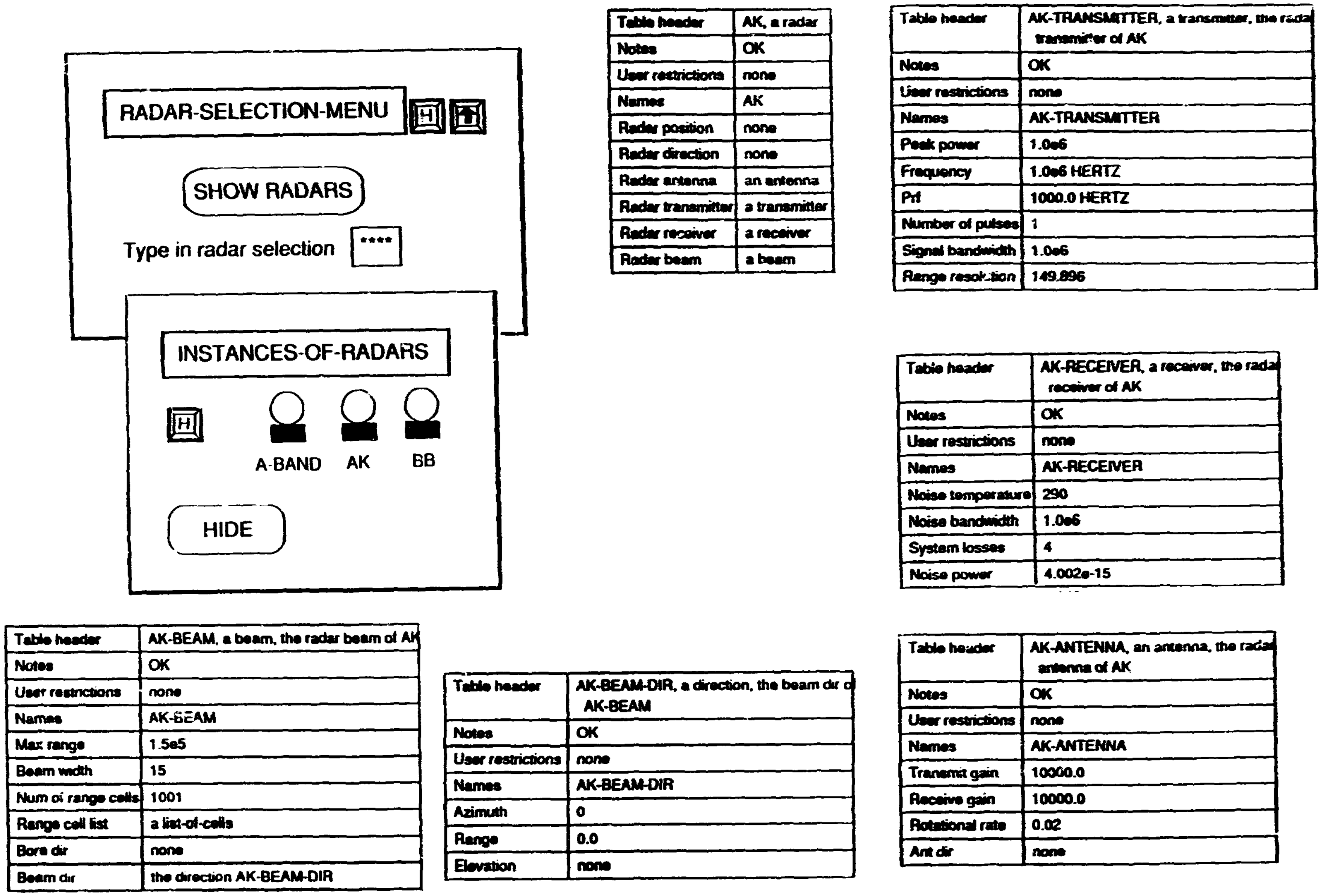


Table header	AK, a radar
Notes	OK
User restrictions	none
Names	AK
Radar position	none
Radar direction	none
Radar antenna	an antenna
Radar transmitter	a transmitter
Radar receiver	a receiver
Radar beam	a beam

Table header	AK-TRANSMITTER, a transmitter, the rada transmitter of AK
Notes	OK
User restrictions	none
Names	AK-TRANSMITTER
Peak power	1.0e6
Frequency	1.0e6 HERTZ
Prf	1000.0 HERTZ
Number of pulses	1
Signal bandwidth	1.0e6
Range resolution	149.896

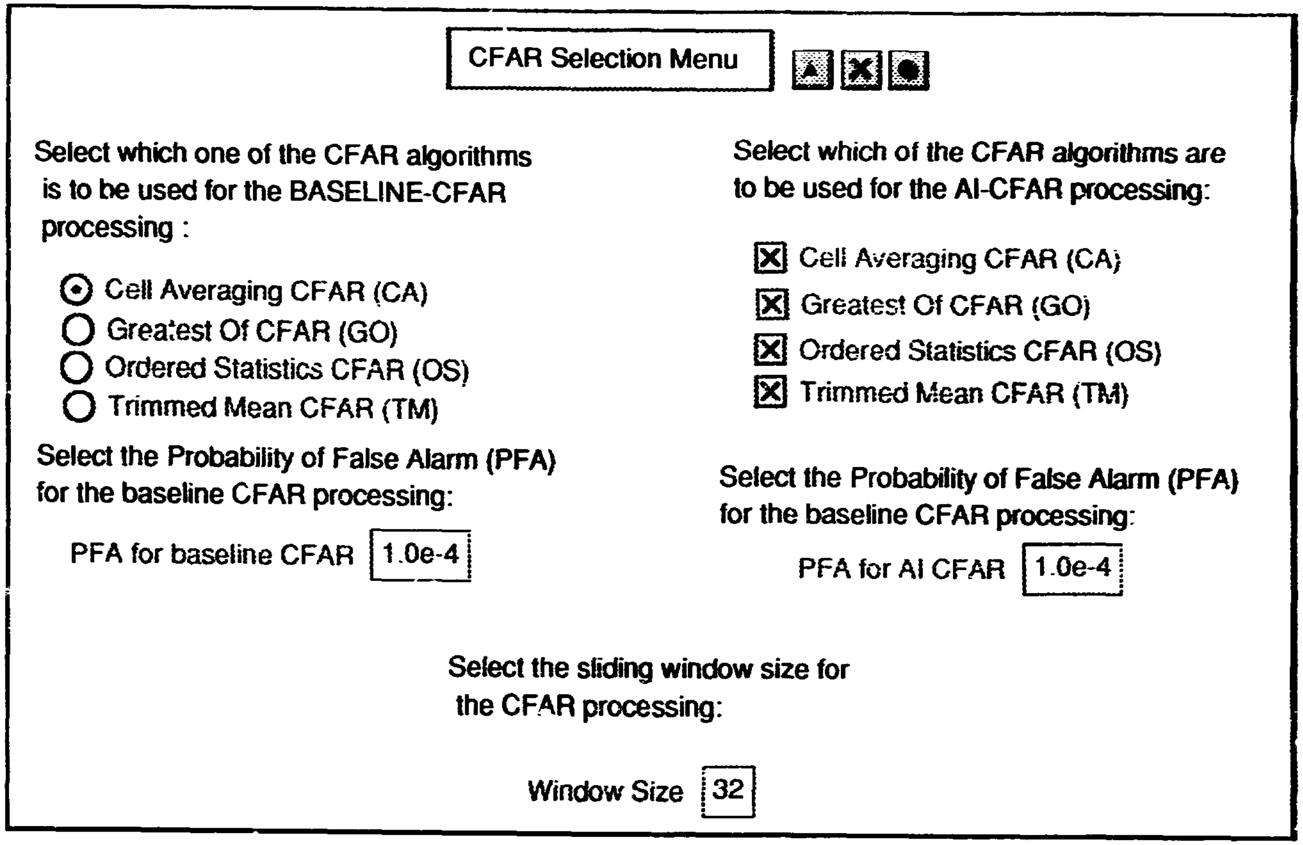
Table header	AK-RECEIVER, a receiver, the rada receiver of AK
Notes	OK
User restrictions	none
Names	AK-RECEIVER
Noise temperature	290
Noise bandwidth	1.0e6
System losses	4
Noise power	4.002e-15

Table header	AK-BEAM, a beam, the radar beam of AK
Notes	OK
User restrictions	none
Names	AK-BEAM
Max range	1.5e5
Beam width	15
Num of range cells	1001
Range cell list	a list-of-cells
Bore dir	none
Beam dir	the direction AK-BEAM-DIR

Table header	AK-BEAM-DIR, a direction, the beam dir of AK-BEAM
Notes	OK
User restrictions	none
Names	AK-BEAM-DIR
Azimuth	0
Range	0.0
Elevation	none

Table header	AK-ANTENNA, an antenna, the rada antenna of AK
Notes	OK
User restrictions	none
Names	AK-ANTENNA
Transmit gain	10000.0
Receive gain	10000.0
Rotational rate	0.02
Ant dir	none

Figure 2-5: Workspaces associated with selecting and changing radar parameters. The user can "click" on table items and edit their values.


CFAR Selection Menu

Select which one of the CFAR algorithms is to be used for the BASELINE-CFAR processing :

[✓ Cell Averaging CFAR (CA)]
Greatest Of CFAR (GO)
Ordered Statistics CFAR (OS)
Trimmed Mean CFAR (TM)

Select the Probability of False Alarm (PFA) for the baseline CFAR processing:

PFA for baseline CFAR 1.0e-4

Select which of the CFAR algorithms are to be used for the AI-CFAR processing:

[✓ Cell Averaging CFAR (CA)]
[✓ Greatest Of CFAR (GO)]
[✓ Ordered Statistics CFAR (OS)]
[✓ Trimmed Mean CFAR (TM)]

Select the Probability of False Alarm (PFA) for the baseline CFAR processing:

PFA for AI CFAR 1.0e-4

Select the sliding window size for the CFAR processing:

Window Size 32

Figure 2-6: Subworkspace for CFAR selection. One CFAR algorithm can be selected as the Baseline. Any or all CFAR algorithms can be selected for ES CFAR operation. The false alarm rate is user selectable.

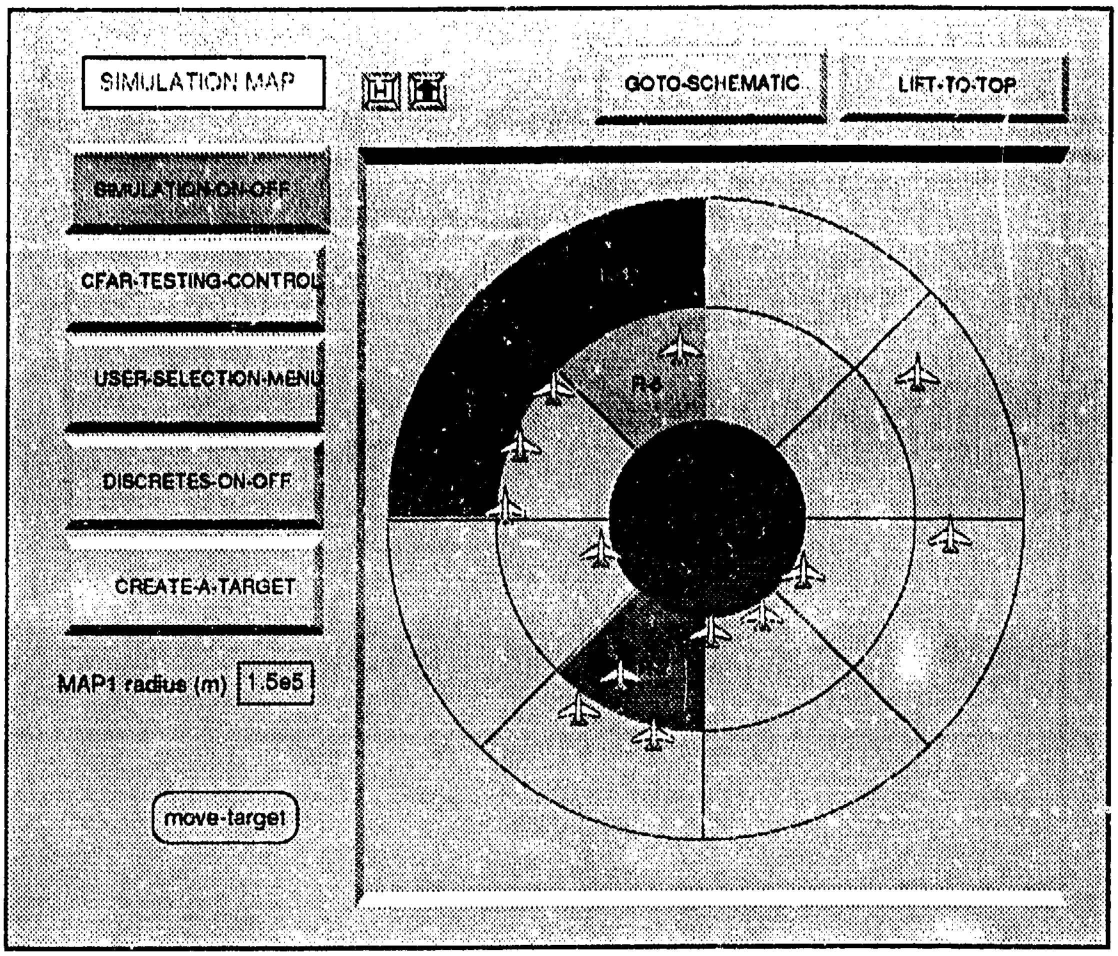


Table header	T1, a target
Notes	TARGET-XXX-33: OK
User restrictions	none
Names	T1
X pos	117
Y pos	-169
Range	65113.132
Azimuth	106.598

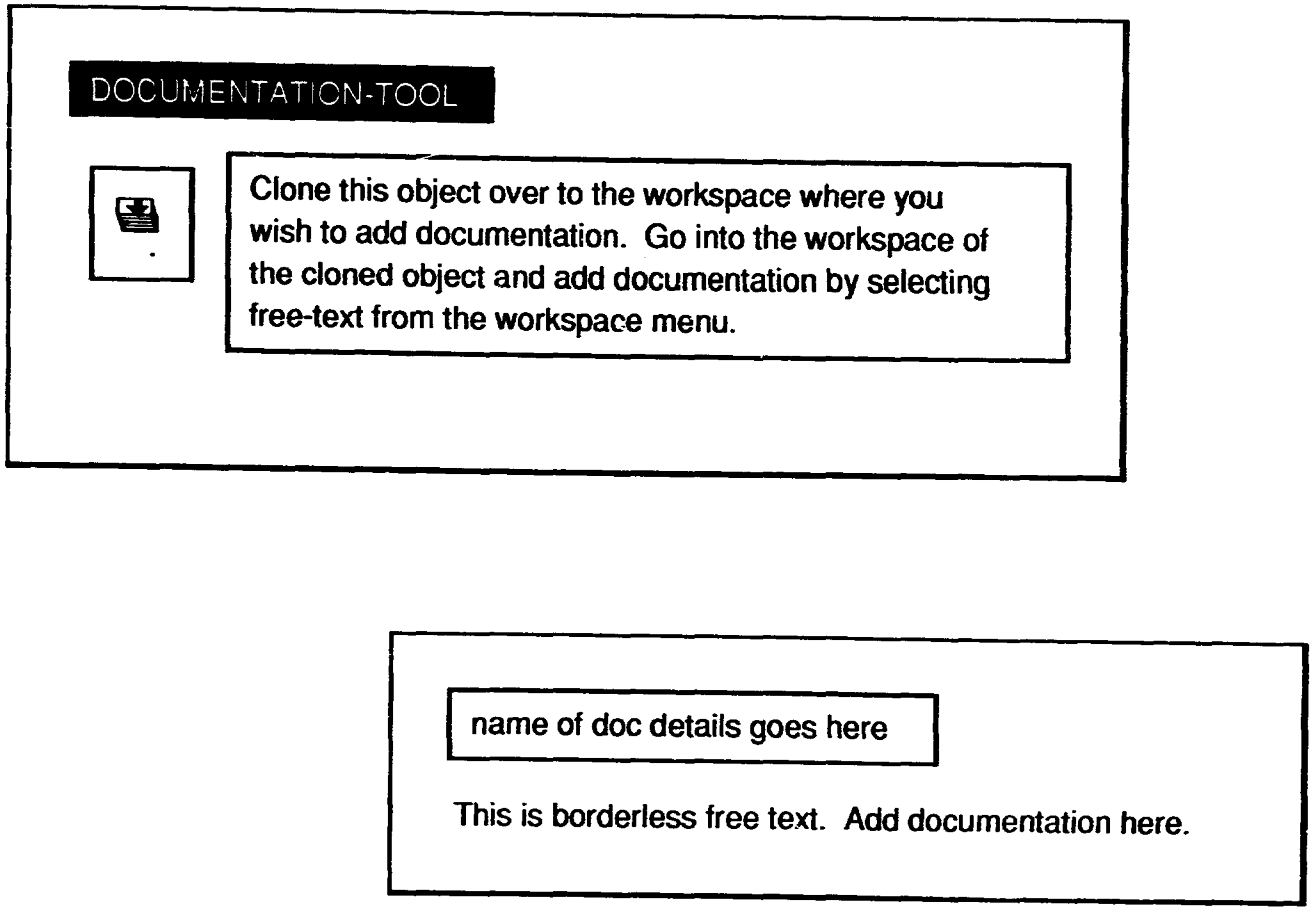
Figure 2-7: Subworkspace for controlling the simulation. Further subworkspaces are accessible by "clicking" on the various action buttons. The table at lower right shows a target's attributes.

procedures by calling the remote procedure SEND-RADAR-PARAMS, which stores the information in a global data structure. The radar parameters are thus made available to the other remote procedures. The user can select clutter regions and set various attributes of the clutter, such as statistical distribution, backscatter coefficient, and distribution shape parameters from the "Simulation Map" subworkspace shown in Figure 2-7. The clutter region information is passed to the remote procedures by calling SEND-CLUTTER-PARAMS for each of the clutter regions. The clutter regions are stored in a linked-list structure. Similarly, the user can create targets and place them relative to the radar location. The list of targets is sent to the remote procedures where it is stored as a linked list. The target attributes that are currently passed are range, azimuth, and radar cross section, as shown in the lower right panel of Figure 2-7.

Workspaces contain logically related objects. It is convenient for the user of the ES CFAR system to also provide documentation on the workspace. A documentation tool, shown in Figure 2-8, was written to help the KB developer document items on the workspace. A sample of documentation on a subworkspace is shown in Figure 2-9.

2.3 Simulated Data

The current set of remote procedures contains functions to generate statistically independent random variates with exponential, Weibull, and log-normal distributions. During the course of the ES CFAR system development, many tests have been run to verify the correct performance of the functions in the remote procedures. The gen_rad_data function generates received power data and stores it in an array for subsequent processing. The data is currently stored in a disc file for interfacing to PV-WAVE for plotting. (With the new version of PV-WAVE, this disc file will be eliminated because disc access is relatively slow for inter-process communication.) Figure 2-10 shows received power versus range for a noise-only environment. The data follows an exponential distribution. Figure 2-11 shows the corresponding received power histogram for the noise-only radial, with amplitude converted to dB. A radial of data with a segment of Weibull-distributed clutter added to the noise is shown in Figure 2-12. The clutter extends from range cell 333 through range cell 665 with a backscatter coefficient of 10^{-7}. A received power histogram of this data is shown in Figure 2-13. In addition to clutter, gen_rad_data

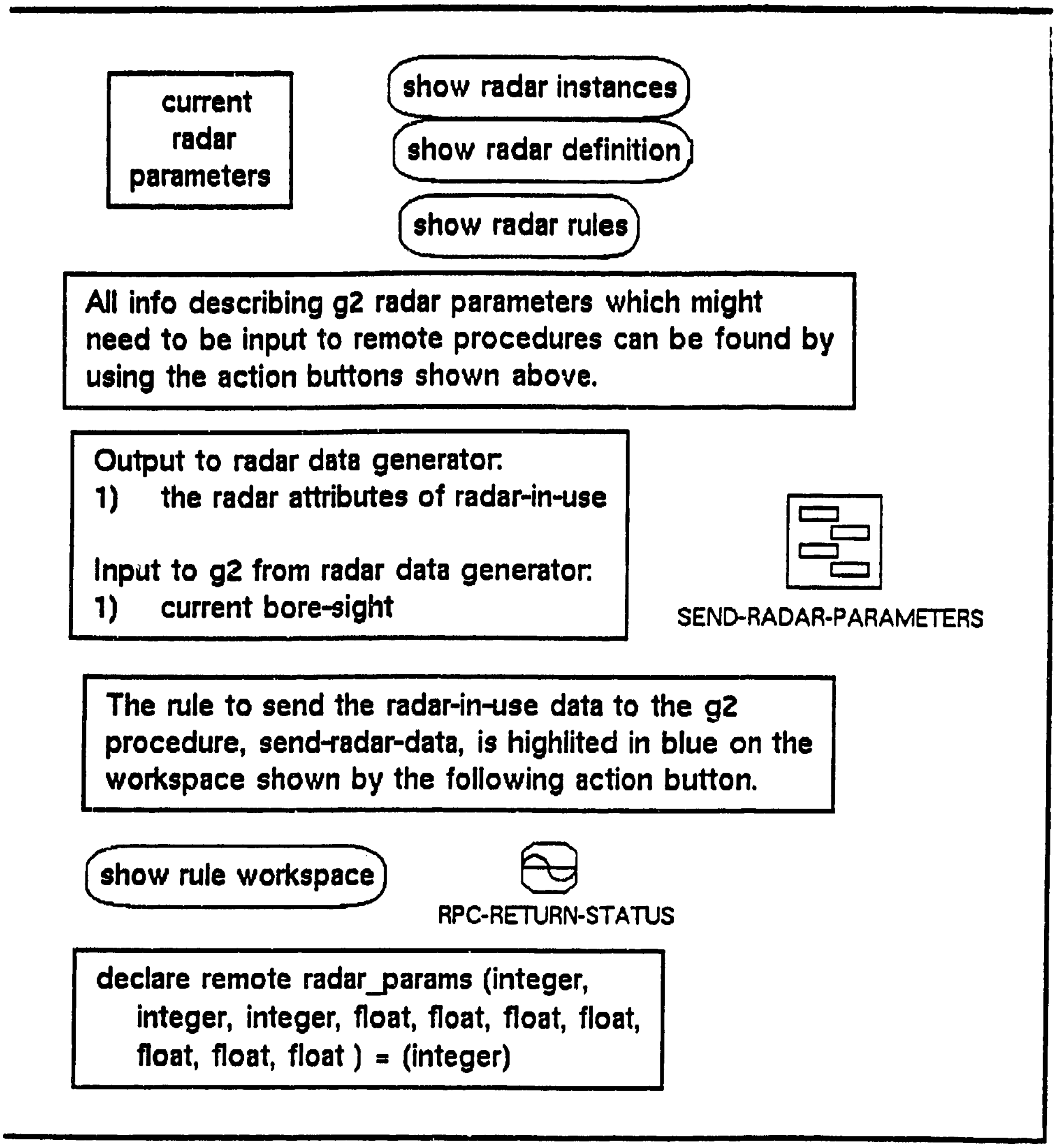

DOCUMENTATION-TOOL

Clone this object over to the workspace where you wish to add documentation. Go into the workspace of the cloned object and add documentation by selecting free-text from the workspace menu.

name of doc details goes here

This is borderless free text. Add documentation here.

Figure 2-8: Documentation Tool workspace. Used to aid the knowledge base developer in documenting a workspace.



current radar parameters

All info describing g2 radar parameters which might need to be input to remote procedures can be found by using the action buttons shown above.

Output to radar data generator:
1) the radar attributes of radar-in-use

Input to g2 from radar data generator:
1) current bore-sight

The rule to send the radar-in-use data to the g2 procedure, send-radar-data, is highlited in blue on the workspace shown by the following action button.

```
declare remote radar_params (integer,
    integer, integer, float, float, float, float,
    float, float, float ) = (integer)
```

Figure 2-9: Subworkspace of current-radar-parameters, with associated documentation and links to other related workspaces.

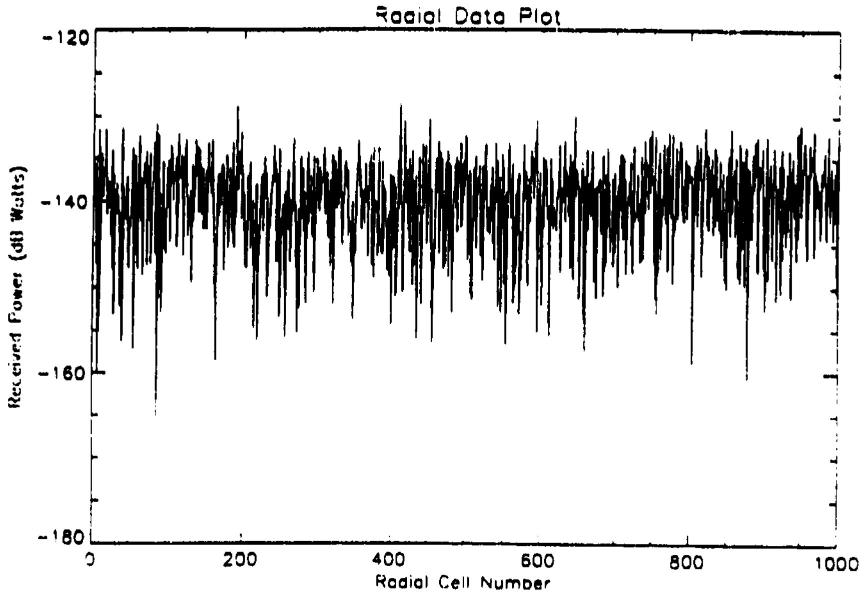


Figure 2-10: Representative noise-only radial of radar received power.

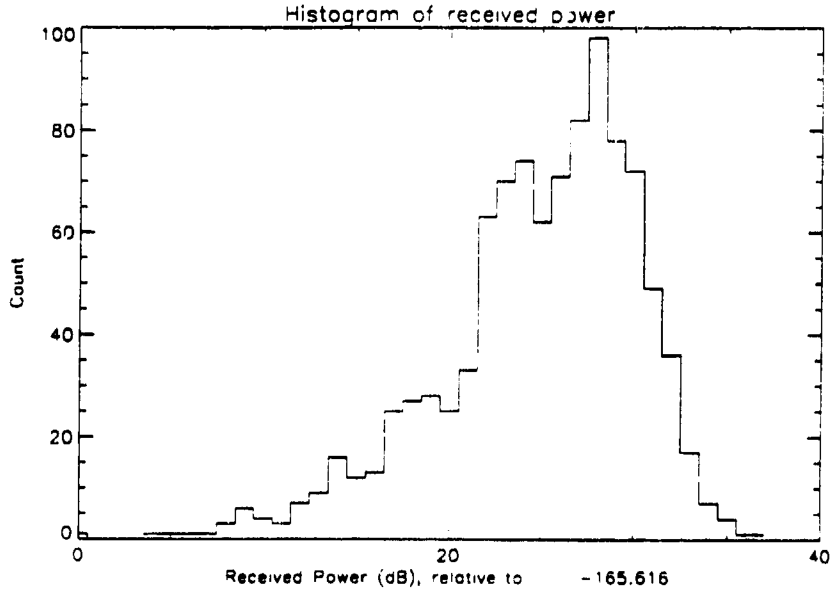


Figure 2-11: Histogram of received power in noise-only radial.

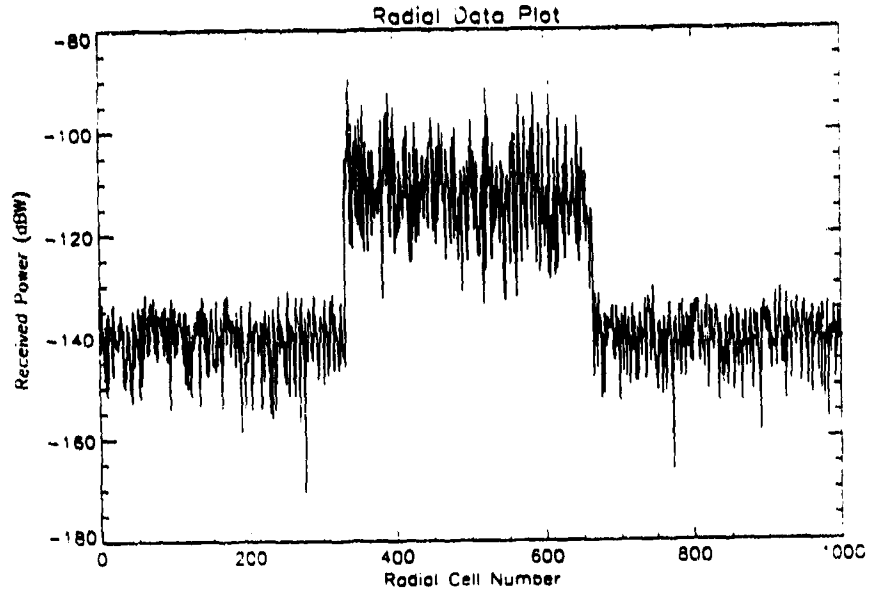


Figure 2-12: Radial of received power including noise and a segment of Weibull clutter. The clutter extends from range cell 333 to range cell 665.

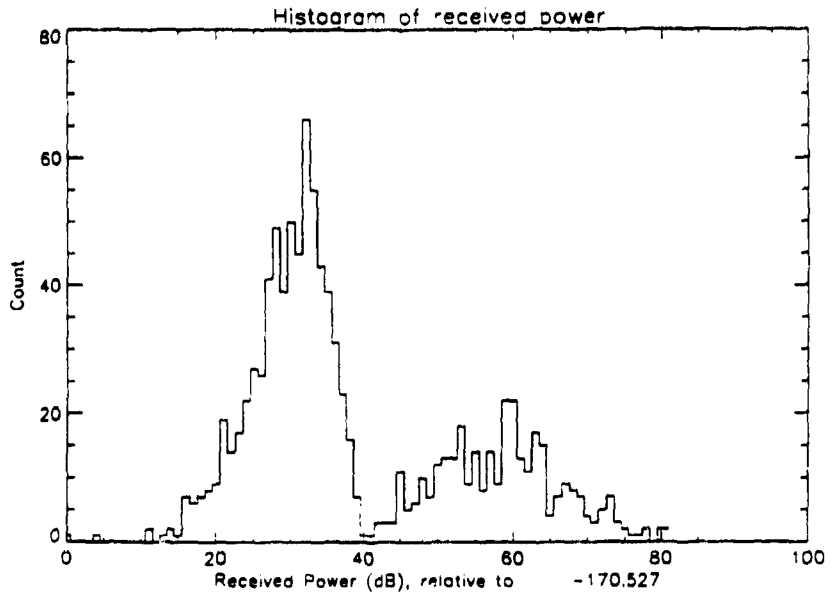


Figure 2-13: Histogram of radial con aining noise and a Weibull clutter segment.

can add targets and discretes. This is illustrated in Figure 2-14. The largest return in Figure 2-14, at range cell 400, is the target. Additional high values of received power are mainly discretes, which have been added every 20 km. Figure 2-15 shows the histogram of the radial with noise, clutter, discretes and a target included.

A number of statistics are computed on the clutter segments. These statistics are sent from the remote procedures to G2 by a call to send-clutter-regions. It is anticipated that these statistics will be used in selecting the CFAR algorithm for a given clutter region. Figure 2-16 shows G2 tables of statistics for two clutter segments. The statistics are the mean, median, minimum, maximum, standard deviation, skewness, kurtosis, and the autocorrelation at lags (delays) 1, 2 and 3.

2.4 Implementation of CFAR Algorithms

Four CFAR algorithms have been selected for use in the ES CFAR knowledge base: Cell averaging (CA), greatest of (GO), ordered statistics (OS), and trimmed mean (TM). These algorithms were coded in C, using the description in Gandhi and Kassam [1]. The C functions are designed to operate on a sliding window, with the test cell at some location within the sliding window. For the baseline implementation, the calling function moves the sliding window through the radial of data, with the test cell centered in the window. Tests are not performed for the first n/2 range cells or the last n/2 range cells, where n is the number of values used for the sliding window, since a full window is not available for averaging in those cases.

In the AI implementation, the radial is divided into segments of relatively homogeneous clutter, based on the clutter edges. The ES CFAR module receives a list of segments from G2 and then exercises the appropriate CFAR algorithm in a given clutter segment. Since the radial is divided into multiple segments, the baseline method of applying the sliding window would result in excessive gaps in range cell processing at the transition regions between segments. Therefore, a different sliding window method is used for the ES CFAR. In the ES CFAR function, the first range cell is also the first test cell. The sliding window is composed of n range cells at farther ranges as shown in Figure 2-17 (a). The window remains fixed while the test cell advances until the test cell is centered in the window. The window and test cell then advance together, keeping the test cell

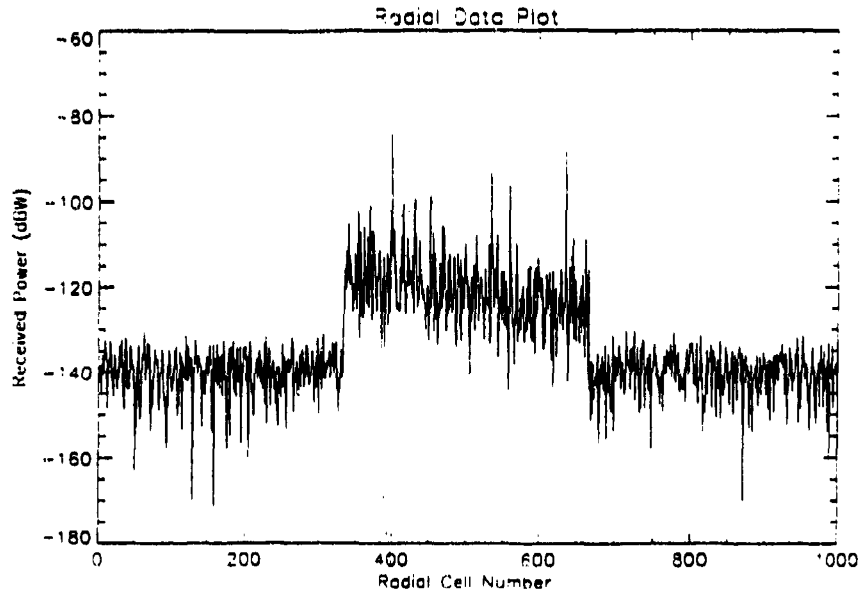


Figure 2-14: Received power for a radial containing noise, clutter, a target, and discretes. The target is located at range cell 400. Discretes are located every 20 range cells.

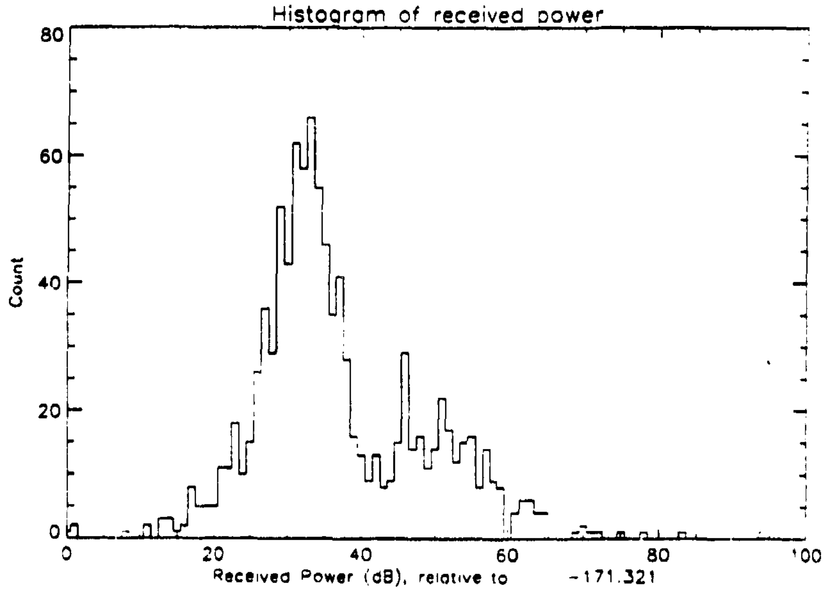
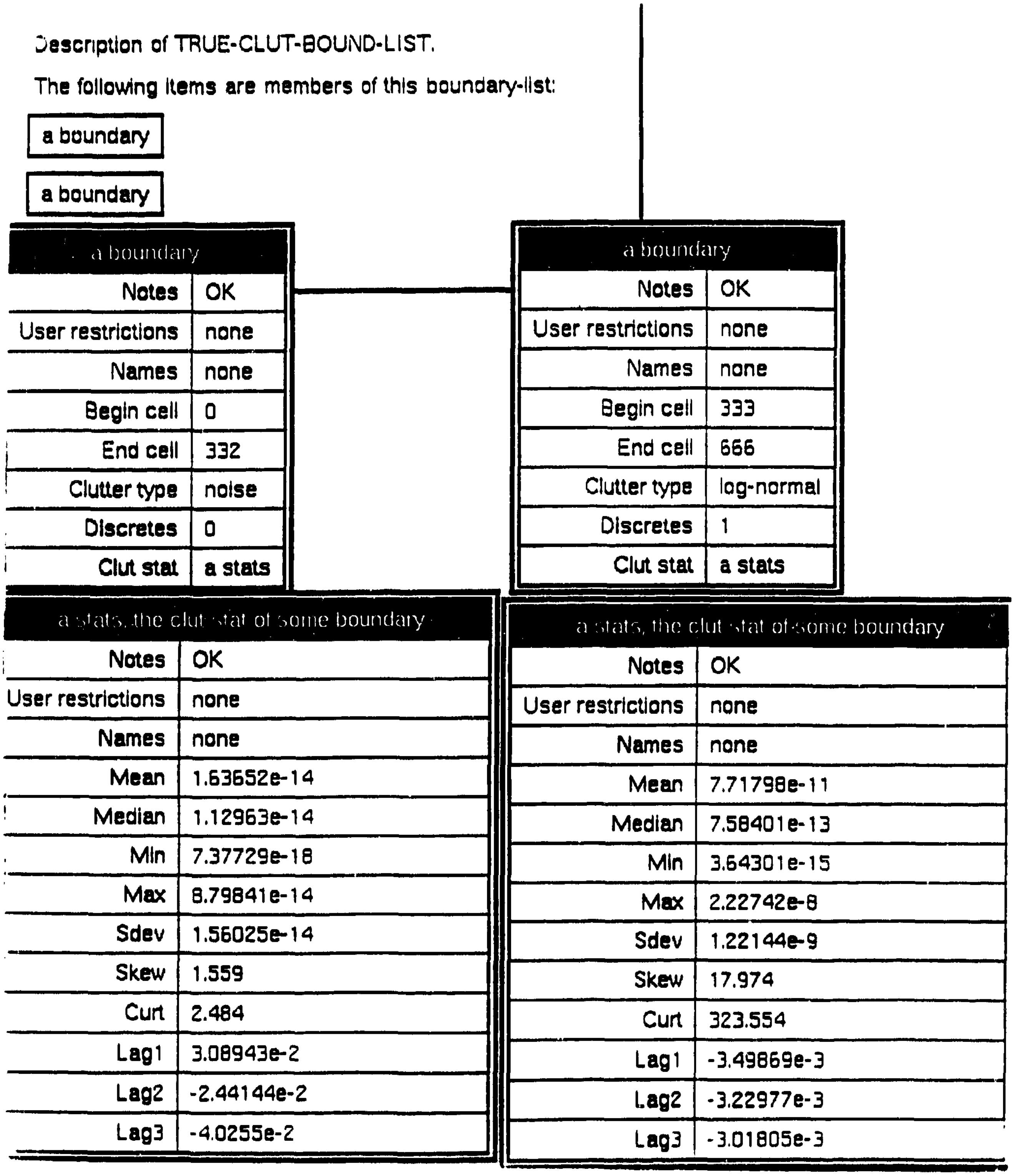


Figure 2-15: Histogram of radial containing noise, clutter, a target, and discretes.



Description of TRUE-CLUT-BOUND-LIST.

The following items are members of this boundary-list:

a boundary

a boundary

a boundary	
Notes	OK
User restrictions	none
Names	none
Begin cell	0
End cell	332
Clutter type	noise
Discretes	0
Clut stat	a stats

a boundary	
Notes	OK
User restrictions	none
Names	none
Begin cell	333
End cell	666
Clutter type	log-normal
Discretes	1
Clut stat	a stats

a stats, the clut stat of some boundary	
Notes	OK
User restrictions	none
Names	none
Mean	1.63652e-14
Median	1.12963e-14
Min	7.37729e-18
Max	8.79841e-14
Sdev	1.56025e-14
Skew	1.559
Curt	2.484
Lag1	3.08943e-2
Lag2	-2.44144e-2
Lag3	-4.0255e-2

a stats, the clut stat of some boundary	
Notes	OK
User restrictions	none
Names	none
Mean	7.71798e-11
Median	7.58401e-13
Min	3.64301e-15
Max	2.22742e-8
Sdev	1.22144e-9
Skew	17.974
Curt	323.554
Lag1	-3.49869e-3
Lag2	-3.22977e-3
Lag3	-3.01805e-3

Figure 2-16: Two instances of the object "boundary". The left represents a segment of noise. The right represents a segment of Weibull clutter, a target, and discretes. The attribute "a stats" is itself an object with attributes that define the statistics of the segment.

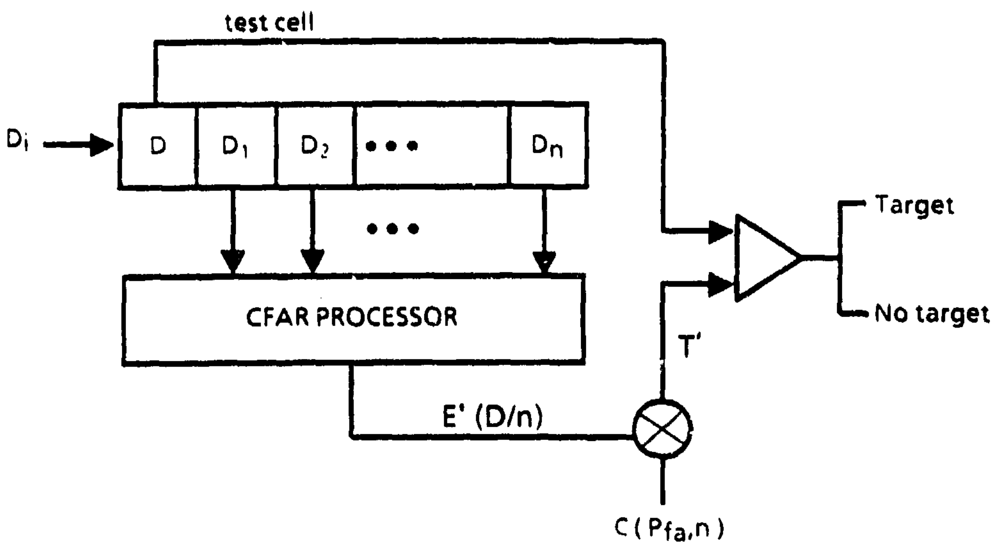


Figure 2-17 (a): Sliding window and test cell for processing the first cell in a clutter segment (expert system path).

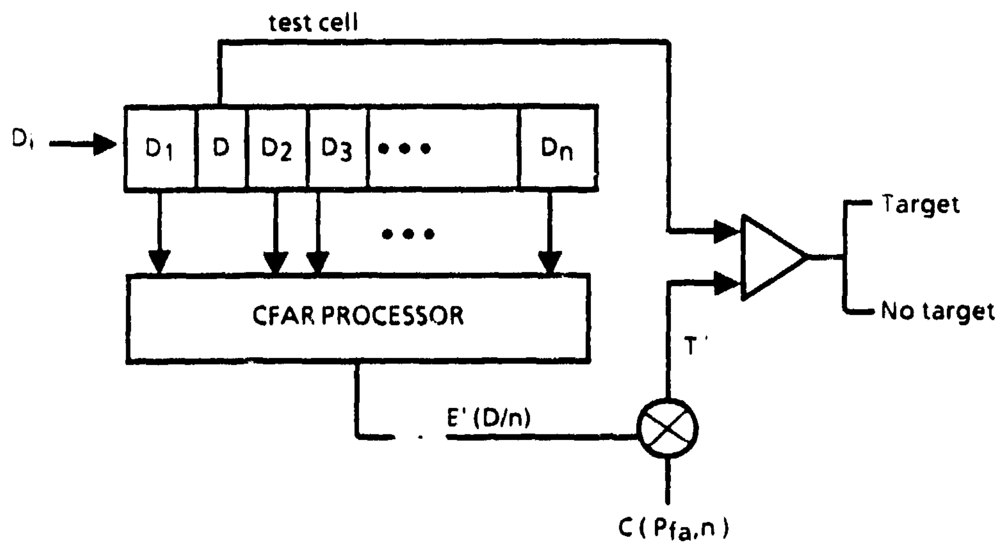


Figure 2-17 (b): Sliding window and test cell for processing the second cell in a clutter segment (expert system path).

centered, until the leading edge of the window reaches the end of the processing segment. At that point the window remains fixed while the test cell advances through the window. The various test cell and window states are illustrated in Figure 2-17 (a-e).

When the first processing segment within the radial is completed, the window is positioned at the start of the next segment with the test cell designated as the first range cell of the new segment. Processing continues as before until all segments in the radial have been processed.

Implementation of the CA and GO algorithms is straightforward. Processing is mainly the accumulation of sums. However, for the OS and TM algorithms the data needs to be sorted and then either a value is picked (OS) or the smallest and largest values are censored (TM). Sorting is accomplished by a heap sort [2] of the pointers to range cells in the radial of data. The data in the radial is therefore undisturbed for processing adjacent test cells.

In making detection decisions, a threshold multiplier is needed for calculation of the threshold from the appropriate test statistic from the data. The threshold multipliers are computed based on the Gaussian assumption as in Gandhi and Kassam [1]. These equations are solved by bisection numerical techniques in a separate program for several values of P_f and the values are inserted in the functions that implement the sliding window. The solution of these equations will be added to the remote procedures of the ES CFAR system and provide threshold multipliers that can be updated for any radial.

2.5 Performance Measures

The performance measures implemented in the current ES CFAR system are the actual P_f and P_d for the baseline and ES CFAR processes. Since the target locations are known for the simulated data, the list of detections can be divided into true detections and false alarms. These numbers are accumulated along with the number of test cells and the actual P_f and P_d are computed as:

P_f = (number of false alarms)/(number of test cells without targets)
P_d = (number of true detections)/(number of actual targets)

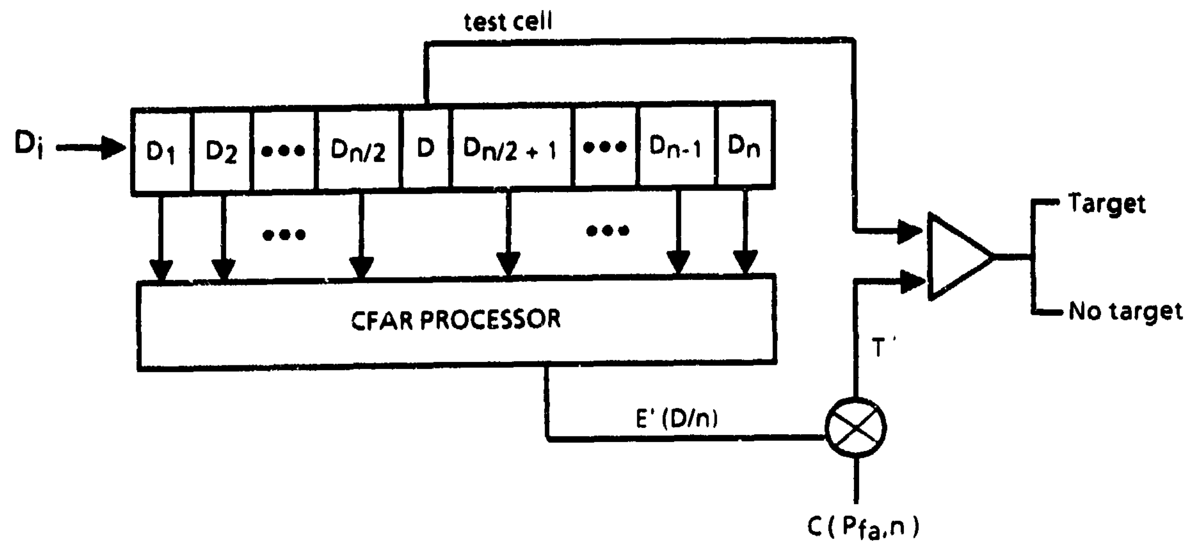


Figure 2-17 (c): Sliding window and test cell when the test cell is far from a clutter boundary (expert system path).

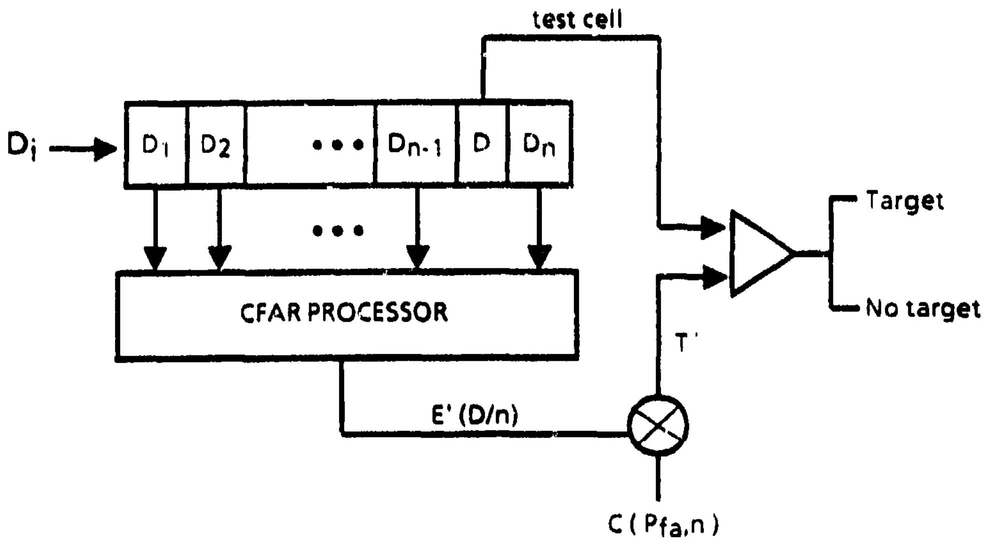


Figure 2-17 (d): Sliding window and test cell for the second to the last test cell in a clutter segment (expert system path).

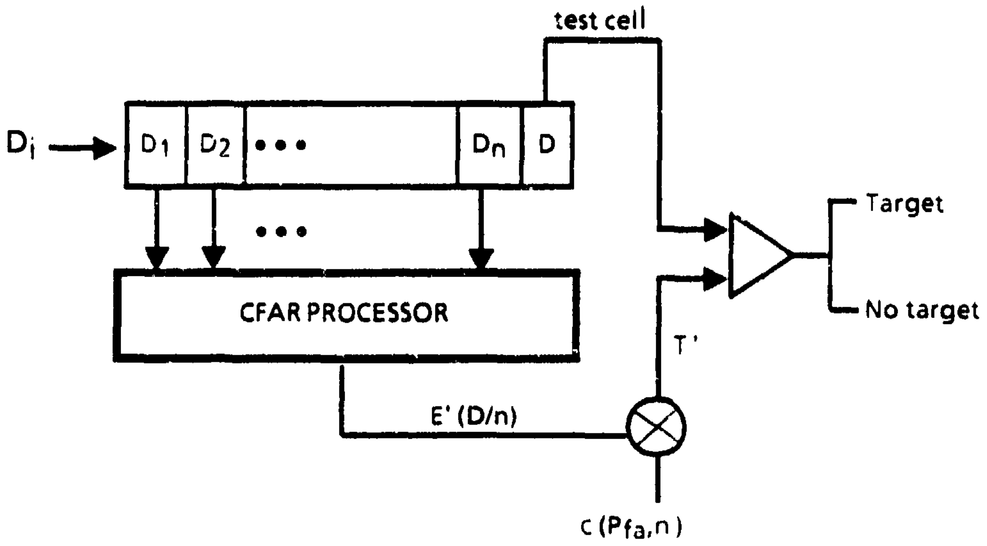


Figure 2-17 (e): Sliding window and test cell for the last test cell in a clutter segment (expert system path).

The P_f and P_d for both the baseline and ES CFAR processes are sent to G2 using GSI, the G2 Standard Interface. GSI is used to build interfaces between G2 and external applications. Currently, the display of P_f and P_d on the time history plot causes data-seeking for values of P_f and P_d. That is, when the display realizes it needs a value for the variable P_f or P_d, it searches for it. Sample time histories of P_f and P_d are shown in section 2.6.

2.6 Testing and Verification of CFAR Algorithms

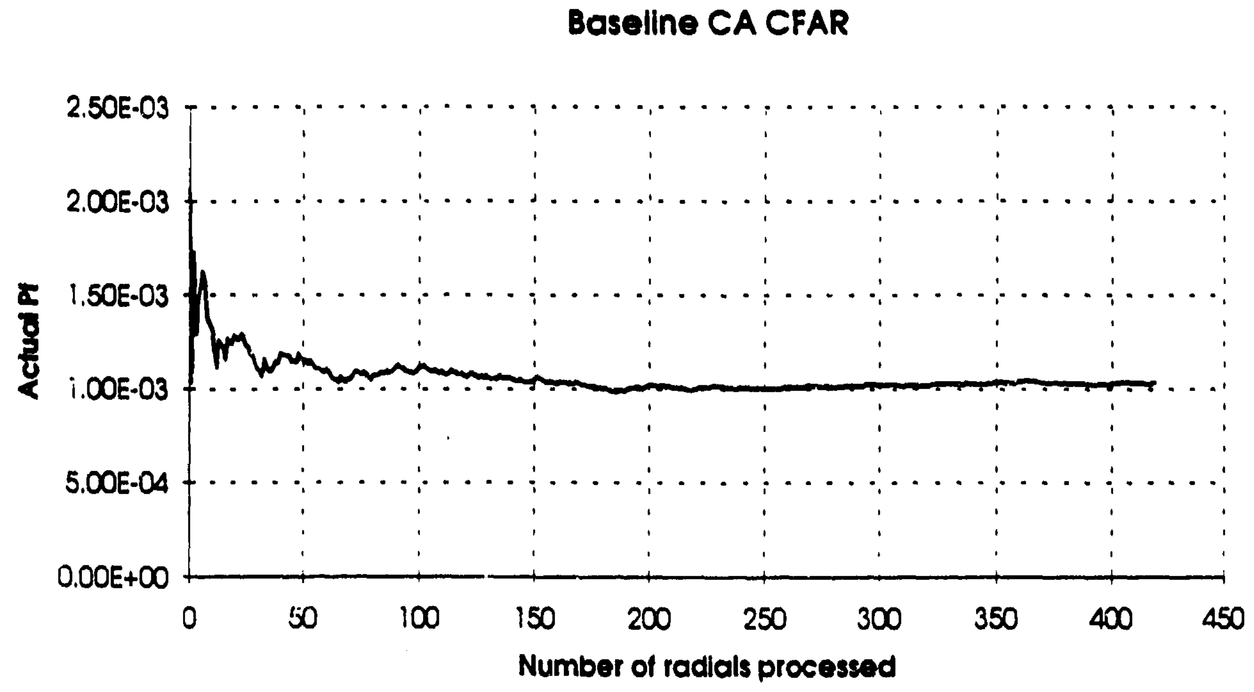
Several different tests were performed on the CFAR algorithms during their development and integration into the ES CFAR system. The CFAR algorithms were developed on a 486 computer using Borland C++. A 486 was used so that the SUN SparcStation could be used for other work that required its graphical capabilities. While a CFAR algorithm was being developed, a test program was used to generate a radial of exponentially distributed clutter to exercise the CFAR algorithm. The test routine computed an actual P_f, which could be compared to the value used to set the threshold for the CFAR routine (ie: the design P_f). When a CFAR routine passed this test, it was then ported to the Sun SparcStation.

To assure that the CFAR routines were correctly incorporated into the ES CFAR system, several test runs were performed. A test case was constructed by inserting a Swerling I target of known radar cross section at a specified range. The test was constructed in this manner to perform a sensitive test of CFAR function logic, threshold calculations, and the distribution of the simulated radar data. To construct the test case, the "AK" radar, which is one of the representative radars included in the ES CFAR system, was selected. For a known SNR, the P_d for a Swerling I target can be computed for exponential clutter or noise using the equations found in Gandhi and Kassam. A SNR of 10 dB was selected since it provides a P_d of approximately 0.5. For the "AK" radar parameters, a target with a radar cross section of -20 dBsm at 74.87 km will give a 10 dB SNR, using the standard radar equation.

To test the CFAR routines in the ES CFAR system, additional rules were added to the knowledge base to conveniently set up the test. These rules generate targets with a user selected RCS and insert them in the list of targets at azimuths that are one beamwidth apart. This setup gives a sufficiently large number of actual targets within a reasonable amount of run time to compute actual P_d. In setting up the test cases, the P_f is set to a rather large value, 0.001, so that many false alarms occur for calculation of actual P_f. In addition, the four CFAR algorithms were tested separately , so rules were added to select only a predetermined CFAR for the ES CFAR path. The same CFAR algorithm was selected for both the baseline and ES CFAR paths. The ES CFAR system was then run long enough to generate at least 300 detected targets.

One of the workspaces in the ES CFAR system allows the user to monitor actual P_f and P_d. The GSI returns values of actual P_f and P_d for both the baseline and the ES CFAR paths. The history-keeping attribute is set so that the previous 200 values can be shown on a graph. Time history performance measures were produced during the test runs by saving values to disk and post-plotting them using Microsoft Excel. These are shown in Figures 2-18 (a-h). The first four figures are the baseline and ES CFAR actual P_f's for CA, GO, OS and TM CFARs. The P_f's for the baseline and ES CFAR paths are slightly different (even though they used the same CFAR algorithm) due to the ES CFAR sliding window procedures discussed in Section 2.4. The last four figures show the actual P_ds for the baseline and ES CFAR paths. Ideally, these P_d's should be identical for the baseline and ES CFAR paths since all of the targets are within the range cells processed by the baseline CFAR. Since both paths operate on the same data, results should be identical.

The operation of the CFARs is a statistical process and the outcome of the application of a CFAR has some uncertainty. The logic of the algorithm can be (and was) tested deterministically. However, to assess actual P_fs and P_ds, statistical tests must be used. The actual P_f's and P_d's are random variables whose distributions are approximately Normal [3] when np (1-p) >9, where n is the number of data points used to estimate the probability and p is the probability being estimated. The uncertainty or standard error of an estimated probability is approximately SQRT [(p(1-p)/n]. A standard statistical test can be used to determine if the P_f and P_d from a test run are acceptably close to their design value (for P_f) or to their calculated value (for P_d). Tables 2-1 and 2-2 show the actual and expected values of P_f and P_d at the end of the test run. The approximate confidence intervals have been computed and are shown in the tables. All estimates of P_f and P_d are within 2 standard errors of their expected values. There is thus no evidence to reject the hypothesis that the



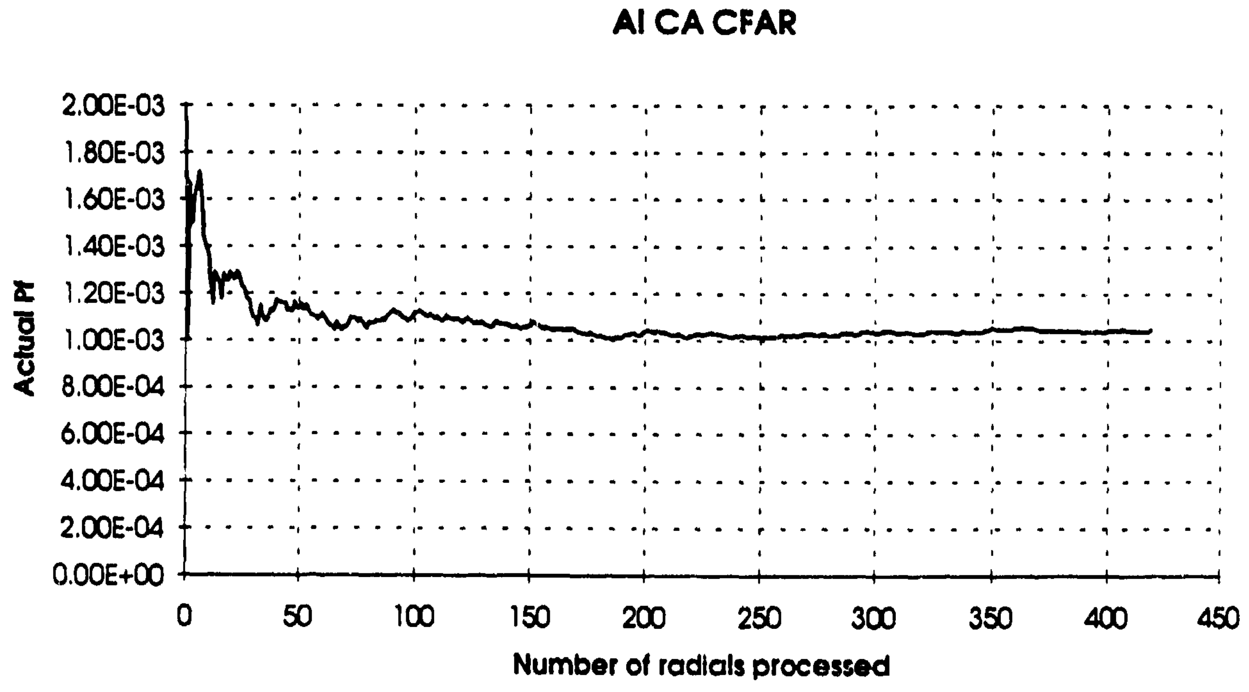
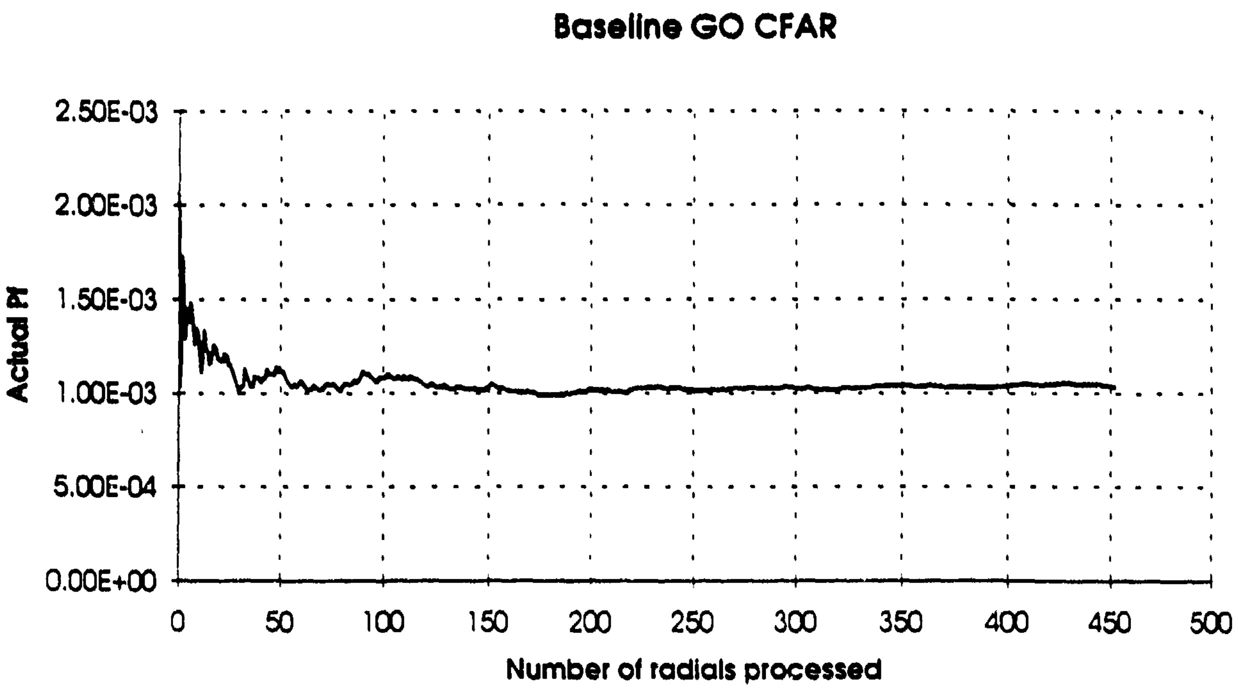


Figure 2-18 (a): Time history comparison of actual P_f for baseline and expert system processing when the CA algorithm is used for both. P_f's vary slightly since the ES processing is different near edges. Results are for exponential (power) background only.



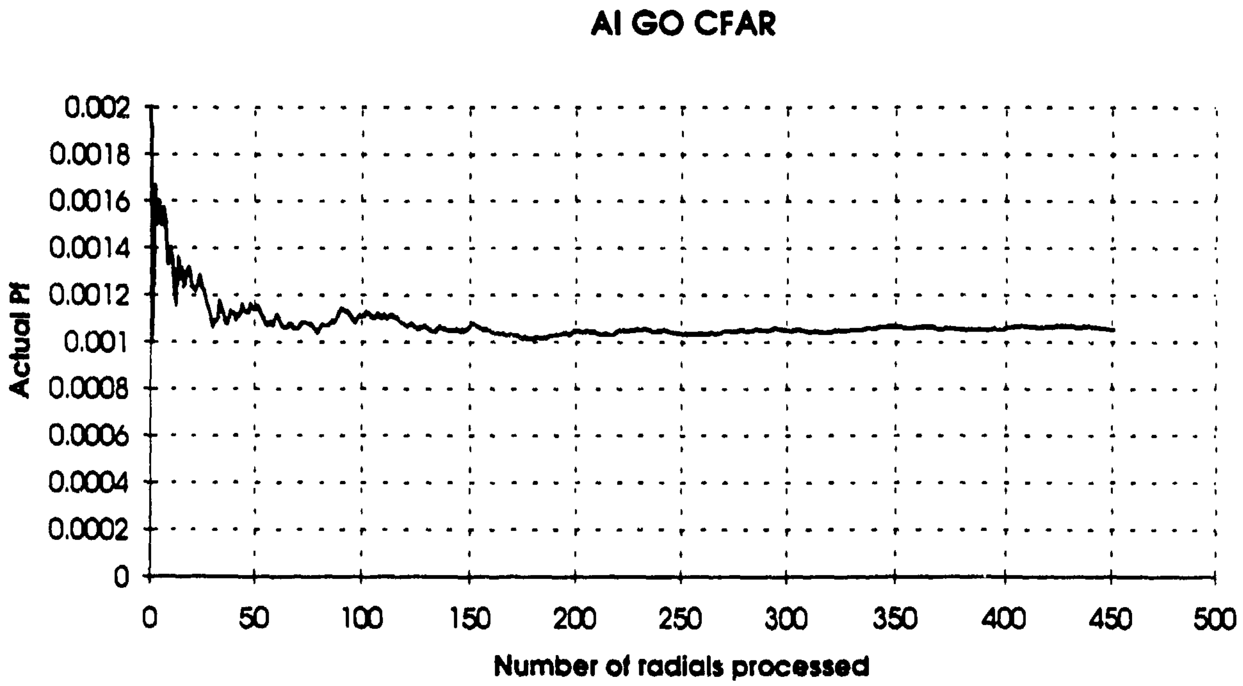
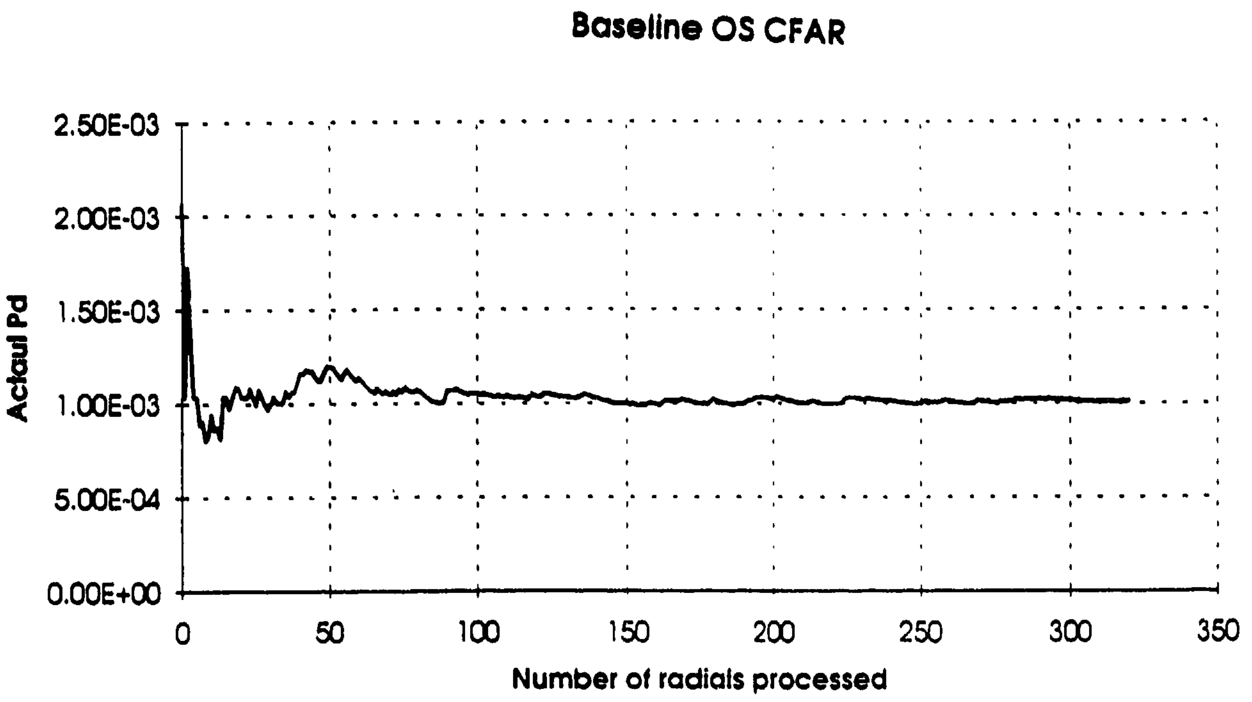


Figure 2-18 (b): Time history comparison of actual P_f for baseline and expert system processing when the GO algorithm is used for both. P_f's vary slightly since the ES processing is different near edges. Results are for exponential (power) background only.



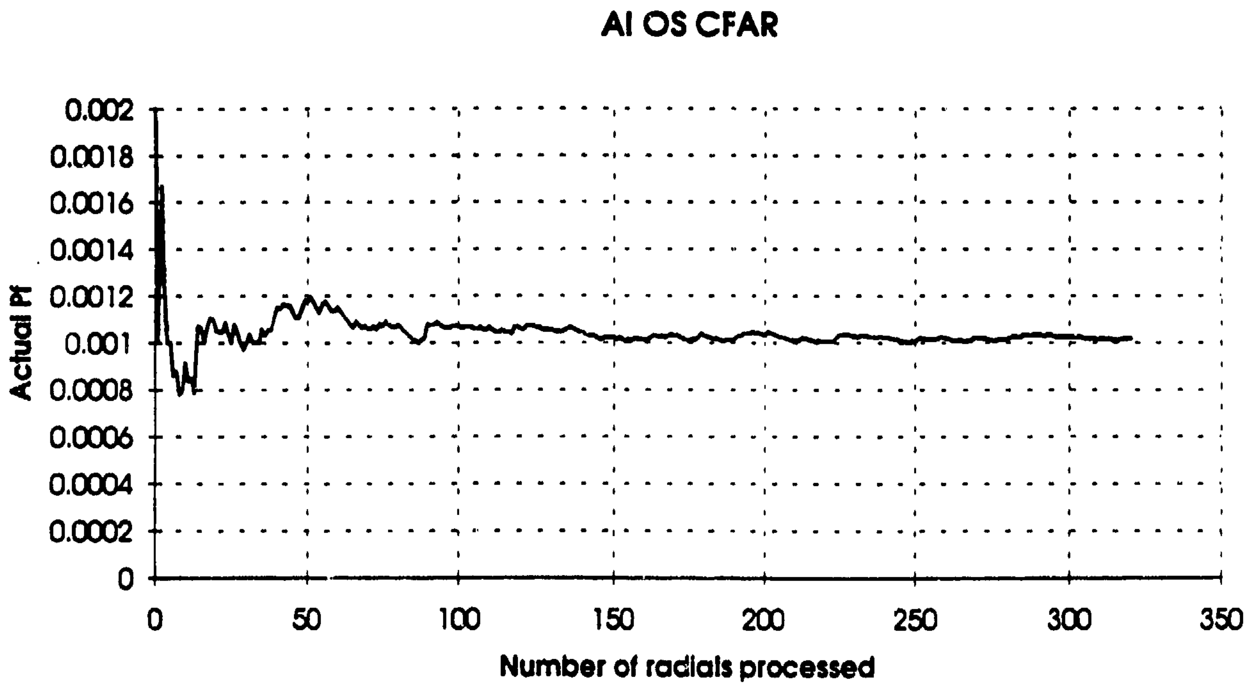
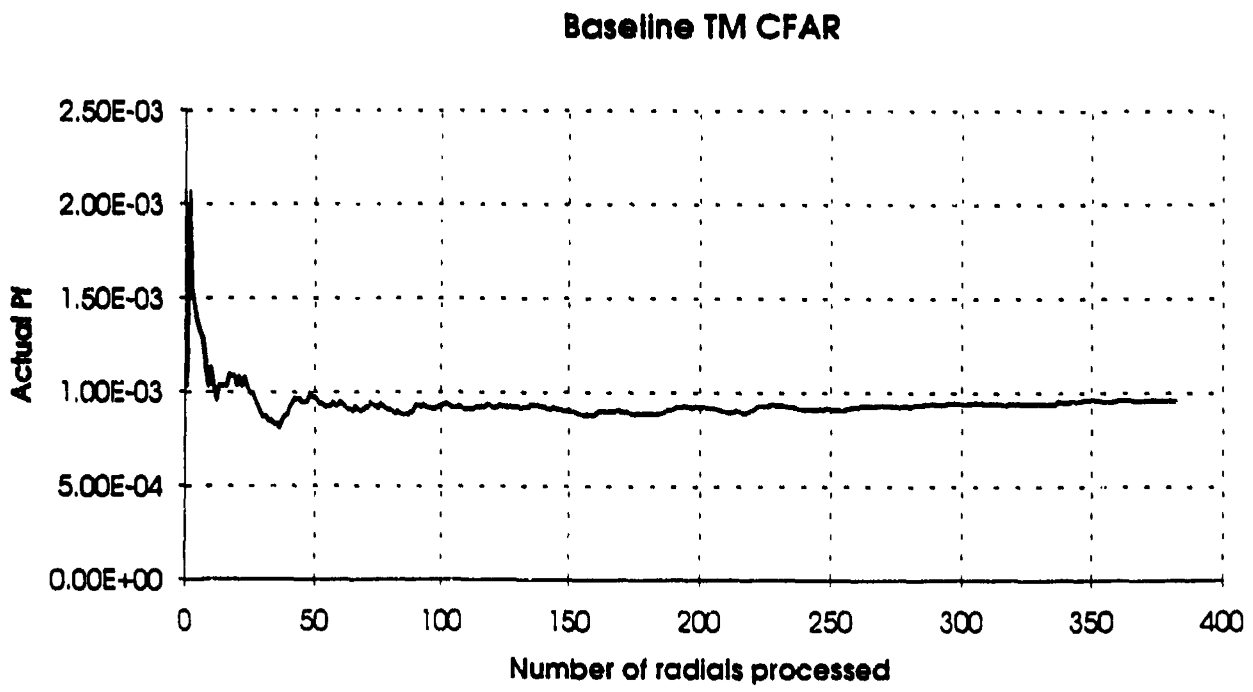


Figure 2-18 (c): Time history comparison of actual P_f for baseline and expert system processing when the OS algorithm is used for both P_f's vary slightly since the ES processing is different near edges. Results are for exponential (power) background only.



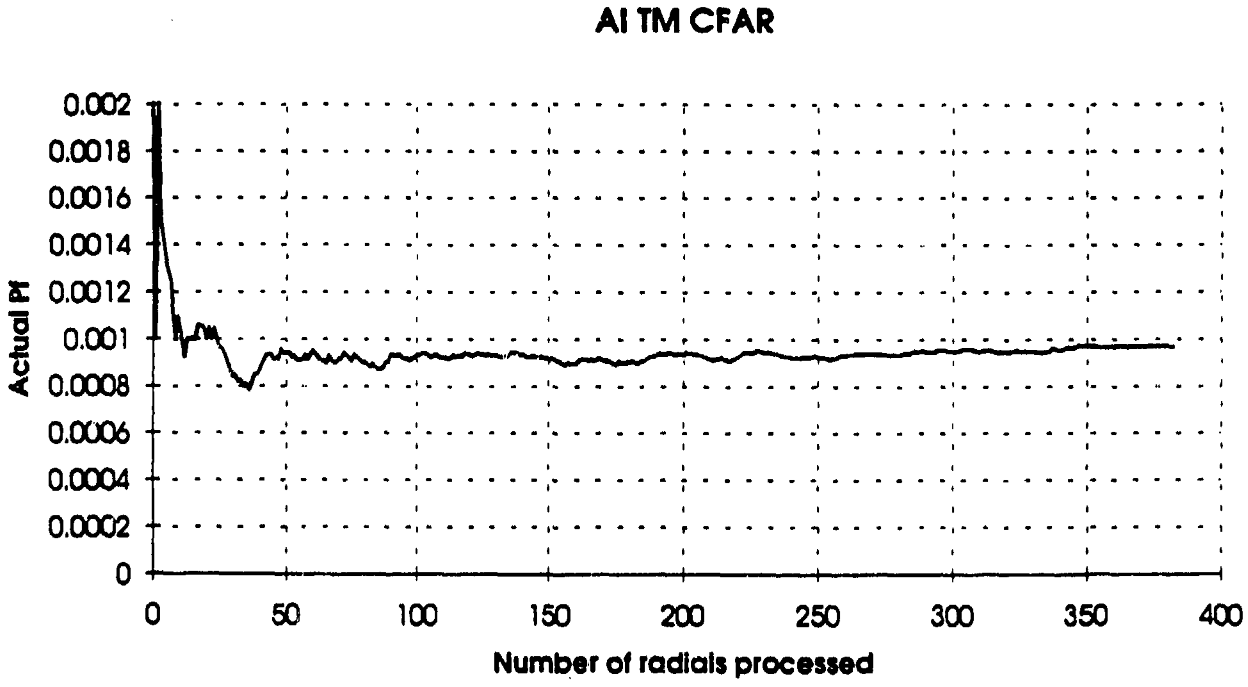
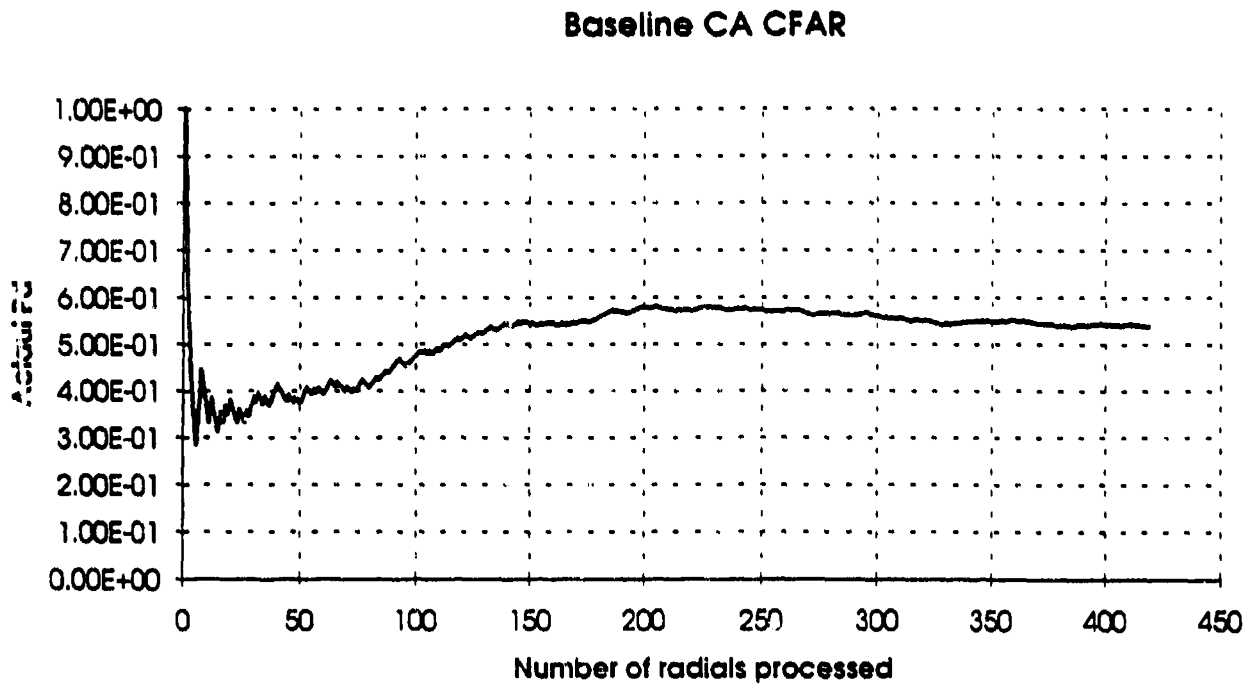


Figure 2-18 (d): Time history comparison of actual P_f for baseline and expert system processing when the TM algorithm is used for both. P_f's vary slightly since the ES processing is different near edges. Results are for exponential (power) background only.



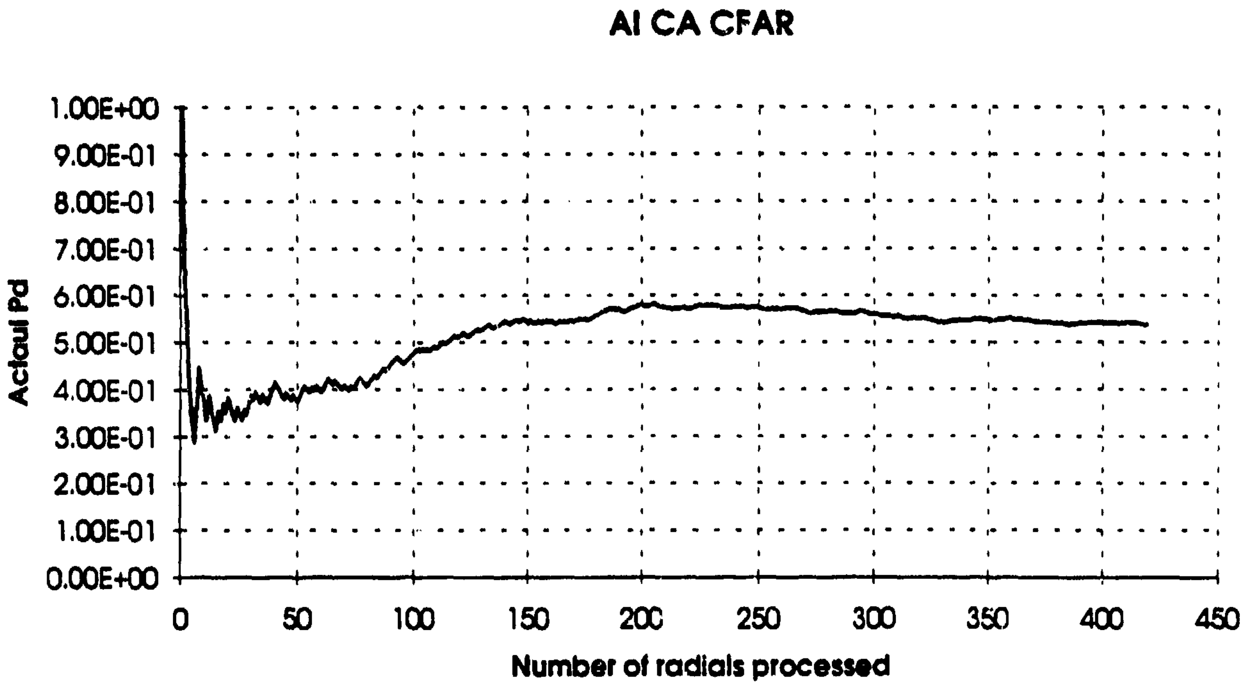
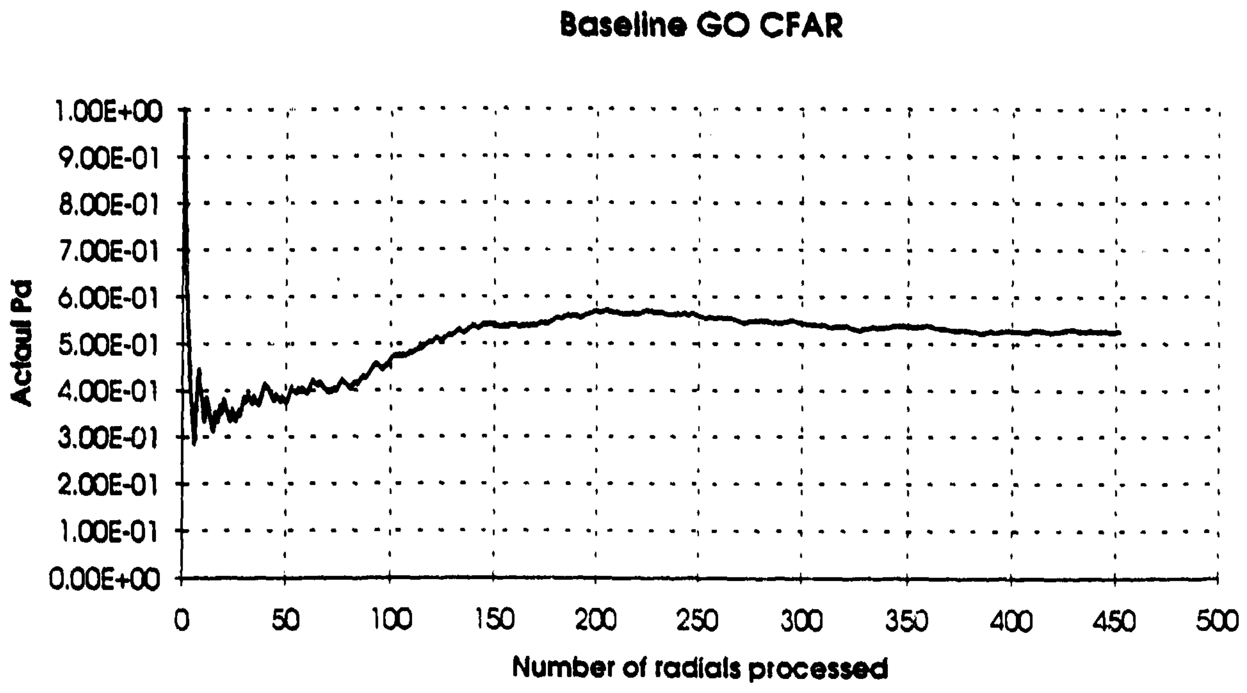


Figure 2-18 (e): Time history comparison of actual P_d for baseline and expert system processing when the CA algorithm is used for both. The RCS of the Swerling I target is set to provide an average SNR of 10 dB.



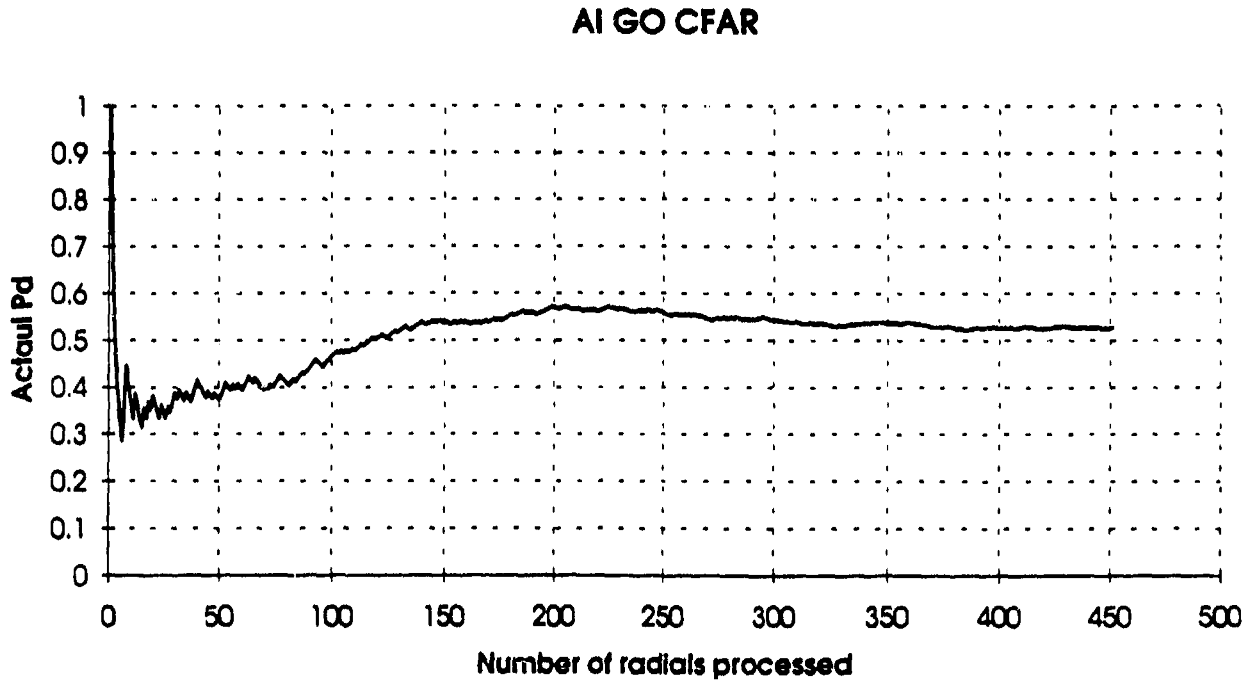
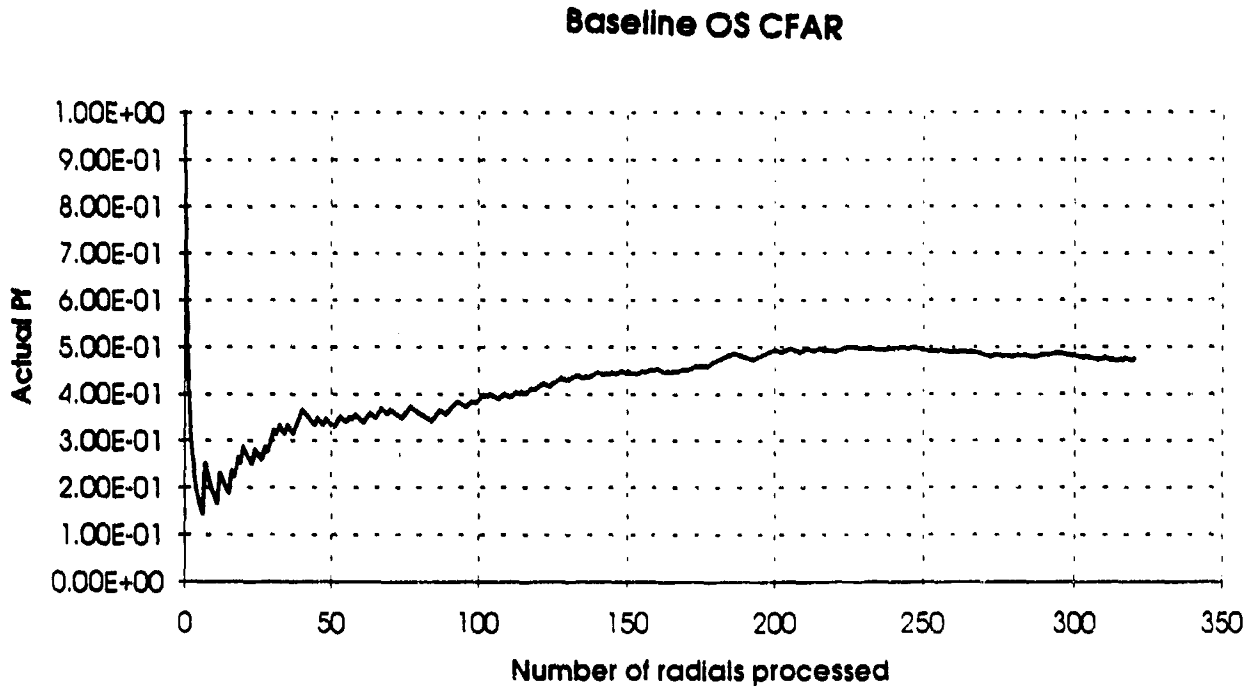


Figure 2-18 (f): Time history comparison of actual P_d for baseline and expert system processing when the GO algorithm is used for both. The RCS of the Swerling I target is set to provide an average SNR of 10 dB.



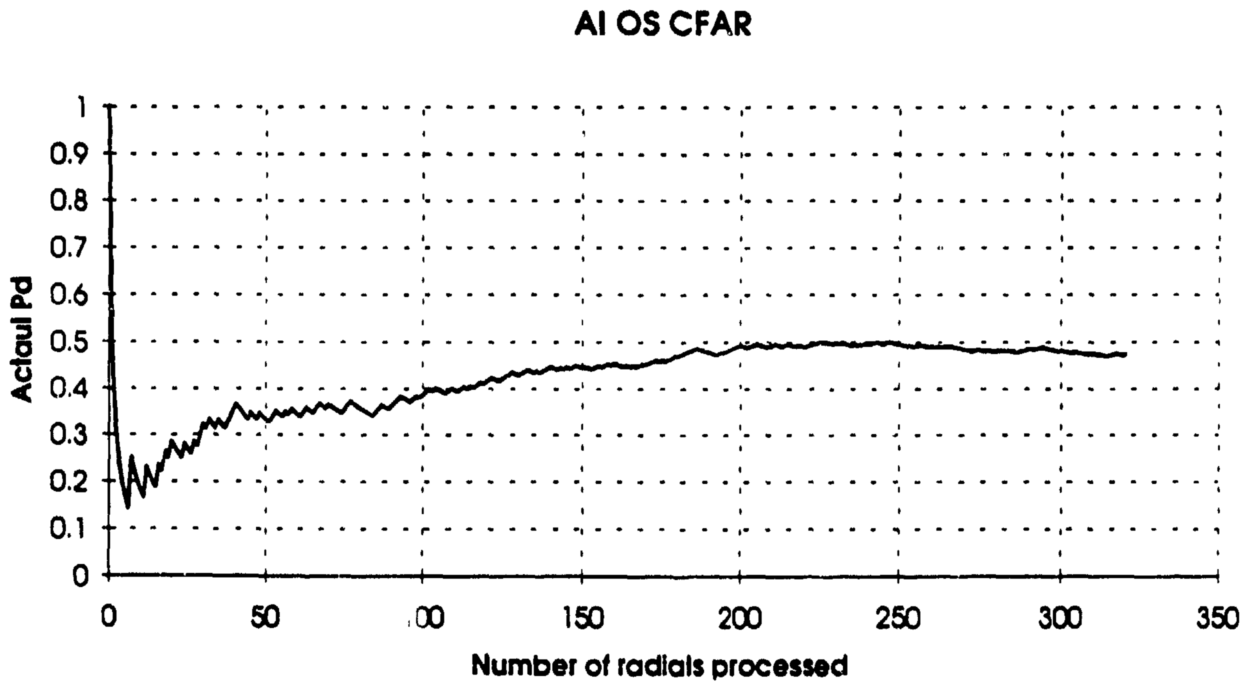
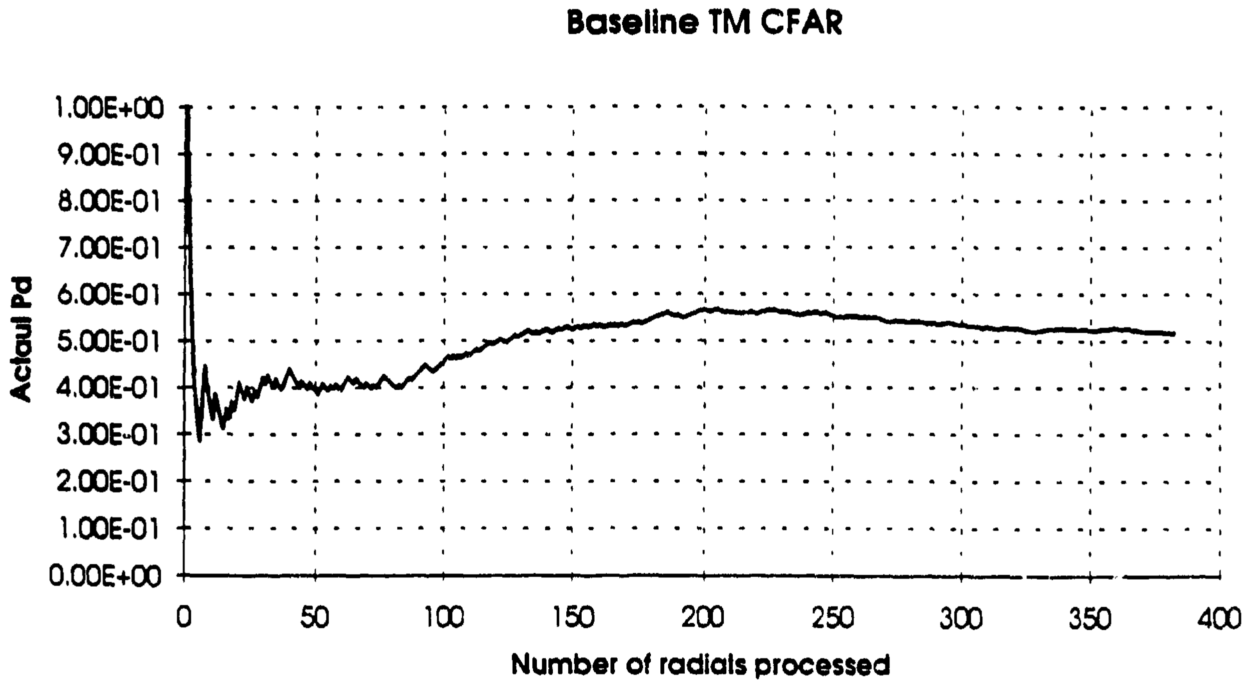


Figure 2-18 (g): Time history comparison of actual P_d for baseline and expert system processing when the OS algorithm is used for both. The RCS of the Swerling I target is set to provide an average SNR of 10dB.



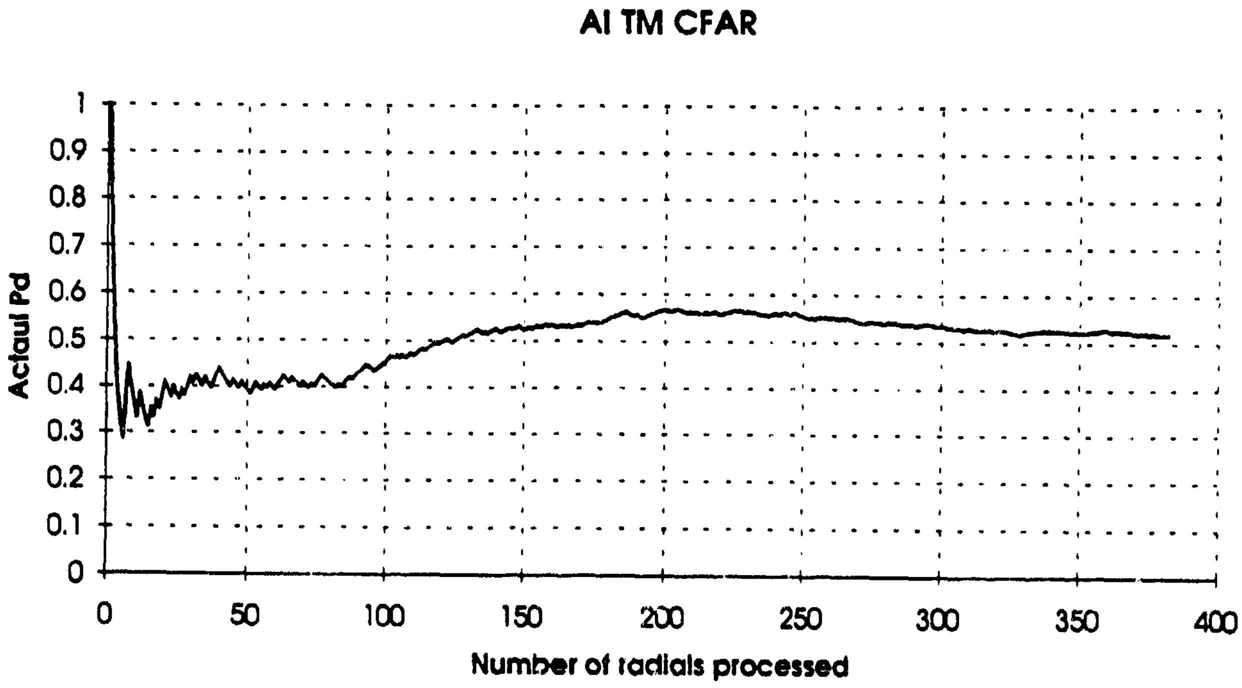


Figure 2-18 (h): Time history comparison of actual P_d for baseline and expert system processing when the TM algorithm is used for both. The RCS of the Swerling I target is set to provide an average SNR of 10 dB.

Table 2-1. False Alarm Probability Comparison.

CFAR	Specified	Baseline	AI-CFAR	# of data	Confidence Interval
CA	0.001	0.00104	0.00103	419	4.88E-05
GO	0.001	0.00105	0.00103	452	4.7E-05
OS	0.001	0.00102	0.00101	321	5.58E-05
TM	0.001	0.00097	0.00096	383	5.11E-05

Table 2-2. Detection Probability Comparison.

CFAR	Calculated	Baseline	AI-CFAR	# of data	Confidence Interval
CA	0.5	0.537	0.537	419	0.024427
GO	0.49	0.527	0.527	452	0.023513
OS	0.42	0.474	0.474	321	0.027548
TM	0.49	0.517	0.517	383	0.025544

estimated P_f and P_d are equal to their expected values. It was therefore concluded that the CFAR algorithms, the exponential random number generation, the operation of the CFAR algorithms through the baseline and ES CFAR paths, and the threshold value calculations were operating correctly.

3.0 CFAR EVALUATION

In parallel to the development of the ES CFAR system, individual CFAR algorithms are being evaluated against a variety of environments. The backgrounds that are being considered include both Gaussian and non-Gaussian as well as both homogeneous and non-homogeneous conditions. In each case, however, the CFAR algorithms are designed for the Gaussian condition. The development of CFAR algorithms for the non-Gaussian case was determined to be outside the scope of this effort. However, some exploratory work was performed in that area.

When using the Gaussian assumption on non-Gaussian data, CFAR performance degrades. Specifically, the actual P_f generally exceeds the design P_f. It is not always possible to force the actual P_f to equal the design P_f by raising the threshold, but even when this is possible detection performance drops dramatically. Nonetheless, some CFAR algorithms do perform better than others in non-Gaussian conditions. This section summarizes the results of the analysis of individual CFAR algorithms in various environments. These results will be used to form some of the rules for the expert system.

3.1 Introduction

In Constant False Alarm Rate (CFAR) radar systems, the aim is to automatically detect a target in a non stationary noise and clutter background while maintaining a constant probability of false alarm. ``Clutter'' refers to any undesired radar signal echo that is reflected back to the receiver by the scatterers that are not of interest to the radar user. Examples of unwanted echoes, or clutter, in radar signal detection are reflections from buildings, sea, rain, birds, chaff etc. The classical detection with a matched filter, followed by a fixed threshold cannot be used for this purpose. This is because when the threshold is fixed, depending on the varying characteristics of the background, either the false alarm probability increases or detection probability decreases intolerably. Therefore, adaptive threshold techniques are needed to maintain a constant false alarm rate (CFAR). For the adaptive threshold setting, information (estimates of the mean level of clutter-plus-noise) is obtained from the local environment of the test cell. This local environment consists of the surrounding range cells and can be defined as a reference window around the radar test cell [4]. In the conventional cell averaging constant false alarm rate detector, (CA-CFAR), the threshold is obtained from the arithmetic mean of the reference window cell samples [5]. When the background is homogeneous and the reference window contains independent and identically distributed (IID) observations governed by the exponential distribution, the CA-CFAR processor maximizes the detection probability, P_d [6, 7]. As the length of the reference window increases, the detection probability, of this system approaches that of the classical Neyman-Pearson optimum detector where the background interference information is known a priori. For the CA-CFAR , the main assumptions are that the reference window is Gaussian, homogeneous and the statistics of interference in the reference window cells are the same as the statistics of interference in the test cell. When this assumption is violated the processor performance degrades significantly. There are two well-known situations in which this assumption does not hold. These are when clutter-edges and multiple target returns are in the reference window.

In the CFAR processor, the square-law detected video range samples are sent serially into a shift register of length (N+1) as shown in Figure 3-1. The leading N/2 samples and the lagging N/2 samples form the reference window cells which are then processed to get the statistic Z, the estimate of the total noise power. To maintain the probability of false alarm, P_f, at a desired constant value when the total background noise is homogeneous, the statistic Z is multiplied by a scale factor T for a given reference window size N. The product TZ is the resulting adaptive threshold. The test cell sample, Y, from the center tap is then compared with this adaptive threshold in order to make a decision. In the design of most systems, the noise samples and target-plus-noise samples are assumed to be Gaussian random variables and the threshold is obtained from the background noise samples which are assumed to be homogeneous, i.e. the samples are independent identically distributed (iid) random variables [5, 7].

Each sample of the received signal from the noisy background has two components, namely the in-phase and the quadrature-phase components. These components are iid Gaussian random variables when the environment is Gaussian. At the square-law detector in the receiver, the square of each component of the received signal samples are summed. The noise sample X_i in the reference window is obtained as

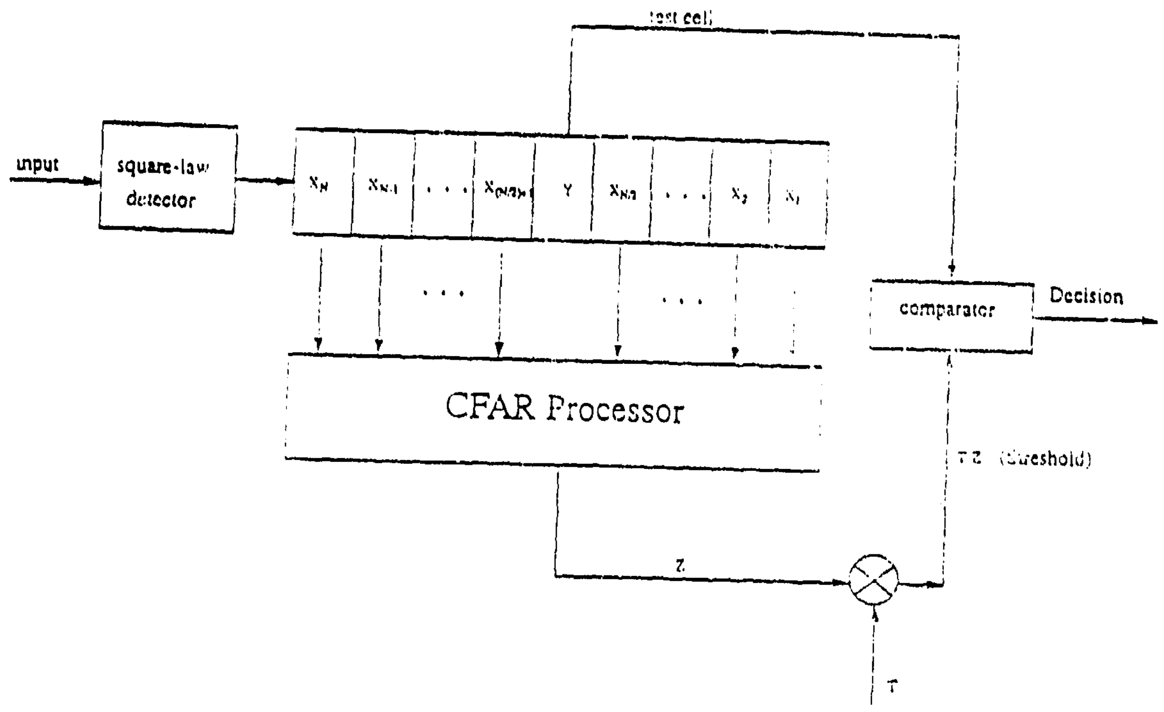


Figure 3-1: A typical CFAR processor configuration.

$$X_i = V_{i1}^2 + V_{i2}^2 \qquad i = 1,2,\ldots,n \tag{3-1}$$

where V_{i1} and V_{i2} are the two components of the returned signal and are independent zero mean Gaussian random variables with the same variance of σ_0^2. In this case, X_i is an exponentially distributed random variable with a single parameter $2\sigma_0^2$.

The tes' cell sample, Y, is also a Gaussian random variable and is defined as:

$$Y = \begin{cases} (V_1 + Y_1)^2 + (V_2 + Y_2)^2, & \text{when there is a target} \\ V_1^2 + V_2^2 & \text{, when there is no target} \end{cases} \tag{3-2}$$

where V_1 and V_2 are noise sample components, Y_1 and Y_2 are target sample components, and all variables are zero-mean Gaussian random variables with variances σ_0^2 for noise sample components and σ_1^2 for target sample components. The random variable Y is also exponentially distributed with parameter $2(\sigma_0^2 + \sigma_1^2)$ when there is a target and with parameter $2\sigma_0^2$, when there is no target. All samples (reference window cells and test cell) are independent.

The power of the noise samples in the homogeneous Gaussian environment is equal to $2\sigma_0^2$, that is

$$E[X_i] = 2\sigma_0^2 \tag{3-3}$$

which is the parameter of the exponential distribution. So the CFAR processor estimates the parameter of the exponential distribution and uses this estimate in the decision process. Probability of false alarm and probability of detection are defined as

$$P_F = \Pr(\text{say "H}_1\text{ : there is target"} | \text{"H}_0\text{ : there is no target"}) \tag{3-4}$$

$$P_d = \Pr(\textit{say} \text{ "H}_1\text{ : there is target"} | \text{"H}_1\text{ : there is target"}) \tag{3-5}$$

In general, the noise power level transition from a clear to a clutter environment is not smooth and can be represented by a step function. This type of transition in the reference window is called a clutter edge. Clutter-edges can affect the system performance in two different ways [4,8]: In the first case, if the test cell is in the clear region but some of the reference cells are covered by clutter, then a masking effect arises. The average of the samples from clear and clutter reference cells increases and raises the threshold. This results in reduced detection and false alarm probabilities even though there is a high signal-to-noise ratio (SNR) in the cell of interest. In the second case, the test cell is immersed in the clutter, but some of the reference cells are in the clear region. Then the threshold becomes relatively low for a test cell sample covered by clutter. The false-alarm probability increases intolerably as the discontinuity between the clear area and the clutter area noise power levels increases [4, 8, 9].

When there is another target (known as an interfering target) within the reference cells, the threshold rises and the detection of the target in the test cell (known as the primary target), seriously degrades [10, 11].

Some modifications have been proposed to deal with the problems of the non homogeneous background. These new schemes do not have the most powerful estimation procedures for the homogeneous background, but they are more resistant to "outliers" and are robust in a non homogeneous background. Due to these modifications some performance loss of detection in a homogeneous background is introduced, but the improved detection in a non homogeneous background is obtained when compared with CA-CFAR.

In some of these modified schemes, the reference window is divided into two subwindows, known as the leading and the lagging windows, symmetrically around the test cell. In order to regulate the severe increases in the probability of false alarm in the region of clutter-edges, Hansen [8] has proposed the selection of the greater-of (GO) the sums in the leading and lagging windows for the estimation of noise power. The additional detection loss introduced by CAGO-CFAR in the homogeneous background is found to be quite small (0.1-0.3 dB) [9, 12]. Although CAGO-CFAR outperforms the CA-CFAR in the region of clutter edges, it is unable to resolve the closely spaced targets. Weiss [11] has shown that when there is an interfering target in the reference window with magnitude equal to the primary target, the detection probability of CAGO-CFAR degrades severely. Rohling [13] has demonstrated this result by doing simulation. He has also shown that in the case of different magnitudes, the target with the smaller magnitude is frequently ``masked'' by the other one.

In order to detect two closely spaced targets, Trunk [14] has proposed the selection of the smaller-of (SO) the sums in the leading and lagging windows for the estimation of noise power. He has shown by simulation that this scheme resolves two closely spaced targets very well. Weiss [11] has shown that in a homogeneous background CASO-CFAR has a small detection loss over CA-CFAR as long as the size of the reference window is kept large, e.g., for $P_f = 10^{-6}$ and Swerling I targets, it is 11 dB for N=4 and only 0.7 dB for N=32. However, as shown by Gandhi and Kassam [1] the detection performance of the CASO-CFAR detector degrades considerably if interfering targets are located in both the leading and the lagging windows. This is because at least one interfering target will raise the threshold value which then leads to masking of the primary target. In addition to this, the CASO-CFAR does not maintain a constant false alarm rate at clutter-edges.

As discussed above, the conventional CFAR schemes can not handle all situations occurring in the background. The drawbacks of these schemes motivate the design of new CFAR schemes which should have the advantages of the already known schemes but avoid their disadvantages. In other words the new CFAR schemes should be robust with respect to interfering multiple target signal returns and/or clutter edges.

Recently a CFAR processing scheme which uses ordered statistics has been introduced by Rohling [13]. It is called the ordered statistics CFAR (OS-CFAR) and is considered to be a robust processor.

The OS-CFAR scheme estimates the noise power level by picking the kth smallest sample in the reference window of size N. Rohling [13] has shown that for the exponential noise model, the false alarm probability is independent of the total noise power . Gandhi and Kassam [1] have shown that with a little degradation in detection probability, the OS-CFAR scheme in exponential homogeneous noise background resolves closely spaced targets effectively for k different from the maximum. However, it is unable to prevent excessive false alarm rate at clutter-edges, unless for the threshold estimate it uses the ordered sample near the maximum, that is unless k is very close to N. But in this case the detection performance of the scheme degrades.

In a more generalized OS-CFAR, ordering can be combined with an arithmetic mean. This combined procedure is known as trimmed mean (TM) filtering. CA-CFAR and OS-CFAR are special cases of the general TM-CFAR. In the TM-CFAR, the samples in the reference window are ordered and then trimmed (censored) from both upper and lower ends. The remaining samples are summed. Trimming can be symmetric or asymmetric at the ends. Gandhi and Kassam [1] have analyzed the TM-CFAR processor. It is shown that by choosing appropriate trimming parameters for asymmetric trimming, in regions of clutter transitions, it is possible to have the false alarm rate of TM-CFAR better than that of OS-CFAR for the same number of reference cells, while also maintaining marginally better detection performance in the homogeneous noise background.

3.2 Performance Analysis of CFAR Processors in K-Distributed Clutter

This section investigates the behavior of CA-, CAGO- and OS-CFAR algorithms in non-Gaussian environments, when the algorithms are designed for Gaussian conditions.

The design of a CFAR processor depends on the given specifications of the system. For the CA-CFAR and CAGO-CFAR processors, these specifications include the false alarm rate, P_f, the reference window size, N, and based on these two parameters the corresponding constant scale factor, T. For the OS-CFAR, in addition to P_F and N, the order number of the sorted reference window sample, k, must be specified before calculating the corresponding constant scale factor T. It should be noted that the value of T is different for each CFAR algorithm.

As typical values for the false alarm rate and reference window size, this analysis uses $Pf = 10^{-3}, 10^{-4}, \ldots, 10^{-6}$, N = 16, 24, & 32 for the CA-CFAR and CAGO-CFAR processors and $Pf = 10^{-3}, 10^{-4}, \ldots, 10^{-6}$; N = 16,24, & 32 ; k = (8, 9,...,16), (12, 13,..., 24), (16, 17,..., 32) respectively for OS-CFAR processor. For each case, the corresponding scale factor T is calculated assuming a Gaussian noise environment when a square-law detector is used. The operation of the algorithm in Gaussian environment is verified for each processor. The number of Monte Carlo simulation runs for each case has been selected as $100/P_f$ or $1000/P_f$.

As an example of the non-Gaussian environment, the K-distribution has been selected as the clutter model. In general, the probability density function (pdf) of the K-distribution is given by

$$p(x) = \frac{4c}{\Gamma(\alpha)}(cx)^{\alpha} K_{\alpha-1}(2cx) \quad (0 \leq x < \infty)$$

where

$K_{\alpha}(x)$ is a modified Bessel function

α is the shape parameter

c is the scale parameter

The 'spikiness' of the K-distributed clutter is heavily dependent on the value of the shape parameter α [15, 16, 17, 18]. The value of α generally lies between 0.1 for very spiky clutter and ∞ for overall Rayleigh clutter where its probability density function becomes

$$g(x) = \frac{x}{\theta} e^{-\frac{x^2}{2\theta}} \quad , x > 0 \qquad (3\text{-}6)$$

with parameter θ. In Figure 3-2, the K-distributed pdf is plotted for several values of the shape parameter.

In the Gaussian environment, the scale factors of the CFAR processors have been calculated with the assumption of a square law detector preceding the processor. The envelope magnitude of the received signal has been modeled as having a K-distribution So, in order to have K-distributed samples in the reference window, the receiver needs to use a linear detector. However, a linear detector has a scale factor value for the CFAR processor that is different from the square-law detector.

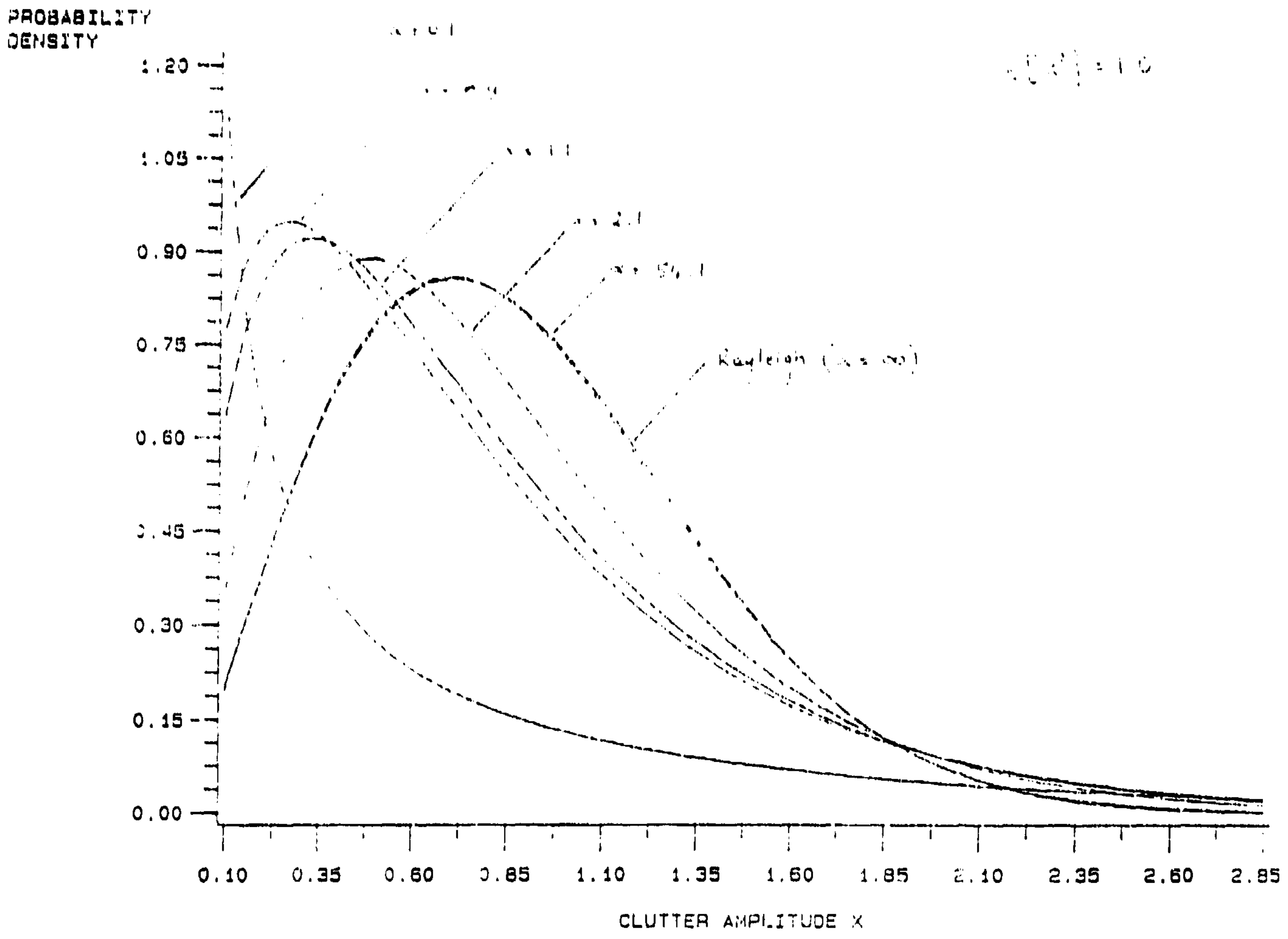


Figure 3-2: The K-distribution pdf for several values of the shape parameter. As the shape parameter increases, the K-distribution approaches the Gaussian condition (Rayleigh amplitude).

In general, it is difficult to analytically calculate the scale factor of a CFAR processor with a linear detector. As an exception, for the OS-CFAR processor, a simple relation exists between scale factor values of linear and square-law detectors which makes it possible to find the scale factor of the linear detector without delving into analytical calculation. For the OS-CFAR detector the relation between these two values is given as

$$T_{lin} \doteq \sqrt{T_{SL}}$$

where T_{lin} is the scale factor for the linear detector and T_{SL} is for the square-law detector.

The OS-CFAR processor was simulated with a linear detector in a K-distributed clutter-only environment while the parameters were set for a Gaussian environment.

The false alarm rate simulation results of OS-CFAR processor in K-distributed homogeneous clutter are shown in Table 3-1.

α	$P_f(100{,}000 runs)$
0.1	0.09017
0.25	0.05056
0.5	0.03034
1.0	0.01720
5.0	0.00457
50	0.00130
500	0.00103

Table 3-1: Simulation results of the OS-CFAR processor in K-distributed clutter. Design $P_f = 10^{-3}$, T = 2.138, N = 32, k = 27.

From this table we observe that the OS-CFAR processor is not able to maintain a constant false alarm rate in a spiky clutter environment ($\alpha \leq 5$).

This simulation result is also in agreement with the result of the work done by Armstrong and Griffiths [18]. In their analysis, they have also assumed a linear detector and have shown that among CA-, CAGO- and OS-CFAR processors, the OS-CFAR processor is the most susceptible of the three to the effects of spiky clutter. In the homogeneous spiky K-distributed clutter environment, the CA-CFAR processor outperforms the other two and the CAGO-CFAR processor lies between the CA and OS processors in terms of performance. But for the ultimate choice of the type of CFAR processor to be used, other factors, such as the susceptibility to clutter edges and interfering targets must be evaluated.

In the rest of this analysis, it is assumed that the CFAR processor is preceded by a square-law detector so that the sample magnitudes are squared before they are sent to the CFAR processor.

In Section 3.2.1, the simulation procedure is explained. In Section 3.2.2 ,the simulation results are given for homogeneous and non homogeneous background conditions. In the homogeneous K-distributed clutter-only environment case, the simulation error Confidence Interval curves are shown for both P_d and P_f versus the shape parameter of the K-distribution.

3.2.1 Simulation Procedure

In the design of a CFAR processor, the threshold multiplier, T, is calculated in terms of the given reference window size, N, false alarm probability, P_f, for the CA-CFAR and the CAGO-CFAR processors and in addition to the parameters given above, the order number, k, for the OS-CFAR processor. A look-up table is made for the threshold multiplier, T, for several values of the parameters given above.

Initially, assume that the clutter returns are much stronger than the Gaussian noise so the samples from background consist only of K-distributed clutter returns.

The in-phase and quadrature-phase components of the complex low-pass K-distributed clutter signal is defined as

$$\tilde{Z} = Z_C + jZ_S \tag{3-7}$$

where the real part, Z_c, is the in-phase component and the imaginary part, Z_s is the quadrature-phase component of the signal. It can be rewritten as

$$\begin{aligned}\tilde{Z} &= (X_C + jX_S)S \\ &= X_C S + jX_S S\end{aligned} \tag{3-8}$$

Then we have

$$Z_C = X_C S \tag{3-9}$$

$$Z_S = X_S S \tag{3-10}$$

where X_c and X_s are Gaussian random variables with zero mean and variance 1. S is a generalized Chi distributed random variable. The probability density function (pdf) of the generalized Chi distributed random variable is obtained from the Gamma pdf by the transformation $s = \sqrt{2y/c}$. The Gamma pdf is given by

$$f(y) = \frac{y^{\alpha-1}e^{-y}}{\Gamma(\alpha)} \quad , \ y > 0 \tag{3-11}$$

where α is the shape parameter and $\Gamma(.)$ is the Gamma function Hence, the pdf of S is found as

$$f(s) = \frac{c^{2\alpha}s^{2\alpha-1}e^{\frac{-s^2c^2}{2}}}{2^{\alpha-1}\Gamma(\alpha)} \quad , s > 0 \tag{3-12}$$

where α is the shape parameter and c is the scale parameter.. The magnitude of the complex variable $\tilde{Z}$ gives

$$|\tilde{Z}| = P \cdot S \tag{3-13}$$

where $P = \sqrt{X_C^2 + X_S^2}$ and has Rayleigh distribution, and S has the generalized Chi distribution. Let's denote the product PS by W, then the density function of K-distributed random variable W is

$$f(w) = \frac{2c}{\Gamma(\alpha)}\left(\frac{cw}{2}\right)^{\alpha} K_{\alpha-1}(cw) \quad , \; w > 0 \tag{3-14}$$

where α is the shape parameter, c is the scale parameter and $K_V(x)$ is the modified Bessel function of the second kind.

As seen above, the magnitude of the clutter is a K-distributed random variable. In CFAR processors the magnitude of the sample is squared and then fed to the shift register.

In the simulation analysis, we have generated the K-distributed random variables, for a given shape parameter α, by generating Rayleigh and generalized Chi distributed random variables and then multiplying them as explained above. One of the features of CFAR structure is that the false alarm probability is independent of the scale parameter of the homogeneous background distribution.

The target is modeled as a slowly fluctuating target, in which case the magnitude of the signal returned from the target fluctuates from sample to sample. It is modeled as a Gaussian random variable. In the low-pass signal representation it becomes a complex Gaussian random variable. At the test cell, the sample is due to the target-plus-clutter return. So

$$\begin{aligned} \tilde{U} &= U_C + jU_S \\ &= (X_{ct} + jX_{st}) + (X_C + jX_S)S \end{aligned} \tag{3-15}$$

where X_{ct} and X_{st} are target signal components and are Gaussian random variables with zero mean and variance $\alpha_1^2; X_c, X_s$ and S, are clutter return components and are the same variables as defined before. Then we have

$$U_c = X_{ct} + X_c S \tag{3-16}$$

$$U_S = X_{st} + X_s S \tag{3-17}$$

In simulation of detection performance, the clutter samples in the reference window are generated from $|\tilde{Z}|^2$ and the test cell sample from $|\tilde{U}|^2$.

Probability of detection is defined for a given power ratio of target and clutter signal returns. The power, P, can be defined as $E\left[|\tilde{U}|^2\right]$ when there is only target signal or when there is only clutter. Hence, we get

$$P_{target} = E\left[|\tilde{U}|^2\right]_{target-only} = 2\sigma_1^2 \tag{3-18}$$

and

$$P_{clutter} = E\left[|\tilde{U}|^2\right]_{clutter-only} = \frac{4\alpha}{c^2} \tag{3-19}$$

Then the signal to clutter power ratio, SCR, is given by

$$SCR = \frac{c^2\sigma_1^2}{2\alpha} \tag{3-20}$$

As a rule of thumb, the number of trials in Monte Carlo simulations is picked as $100/P_f$ but in order to minimize the fluctuations in P_d versus α (shape parameter) plot, the number of trials is 10 times larger, i.e., it is picked as $1000/P_f$.

In the simulation, for each run, define D_i and F_i as

$$D_i = \begin{cases} 1 & \text{, if decision is } H_1 \text{ when } H_1 \text{ is true} \\ 0 & \text{, otherwise} \end{cases} \tag{3-21}$$

and

$$F_i = \begin{cases} 1 & \text{, if decision is } H_1 \text{ when } H_0 \text{ is true} \\ 0 & \text{, otherwise} \end{cases} \tag{3-22}$$

So the unbiased estimates of P_d and P_f are:

$$\hat{P}_d = \left(\frac{1}{n}\right)\sum_{i=1}^{n} D_i \tag{3-23}$$

$$\hat{P}_f = \left(\frac{1}{n}\right) = \sum_{i=1}^{n} F_i \tag{3-24}$$

where n is the total number of trials. Since the true values of P_d and P_f, are unknown, in order to see how good the estimates are for a given n, we have calculated the 95% Confidence Interval. For this the sample variance, $\hat{\sigma}^2_{s_k}$, is obtained by

$$\hat{\sigma}^2_{s_k} = \frac{p_k(1-p_k)}{n} \quad , k = 1,2 \tag{3-25}$$

where $p_1 = \hat{P}_f$ and $p_2 = \hat{P}_d$. Then the 95% Confidence Interval for the estimates $\hat{P}_f$ and $\hat{P}_d$ are approximately given by [19]:

$$(p_k - 2\hat{\sigma}_{s_k}, p_k + 2\hat{\sigma}_{s_k}) \quad , k = 1,2 \tag{3-26}$$

By plotting these confidence intervals we have observed whether the differences in performance among the algorithms are real or could be random variations.

The uniformly distributed random number generator used in the simulations is described by [20]:

$$x_{i+1} = 16807x_i \pmod{2^{31} - 1} \tag{3-27}$$

This generator was used with two different initial seed values. The first seed value was picked to be an odd number. The simulation was run and at the end of the simulation, the last value of the seed was stored. Then the simulation was repeated for the same algorithm parameters with this new initial seed value. The simulation results have been analyzed to determine if there is any significant difference between performance.

Another generator was also used which is described as

$$x_{i+1} = 950706376x_i \pmod{2^{31} - 1} \tag{3-28}$$

In all cases, the variations in the simulation results were found to be within the confidence intervals given before.

When the power level of the background Gaussian noise is comparable to that of the clutter, the noise+clutter samples are generated by adding the in-phase and quadrature-phase components of a complex valued Gaussian random variable to the respective components of a complex valued K-distributed random variable. The test cell sample with a target can be generated by adding a complex valued Gaussian random variable to the complex valued random variable given above. If $P_{ptarget}$, $P_{itarget}$, $P_{clutter}$ and P_{noise} are the mean-square power levels of the primary target, interfering target, clutter, and noise respectively, then we define signal (primary target) to clutter power ratio, SCR, clutter to noise power ratio, CNR, and signal to interfering target power ratio, SIR, as:

$$SNR = P_{pt\arg et} / P_{clutter} \tag{3-29}$$

$$CNR = P_{clutter} / P_{noise} \tag{3-30}$$

$$SIR = P_{pt\arg et} / P_{it\arg et} \tag{3-31}$$

In the simulations the system parameter values have been selected as

$$P_f = 10^{-3} \tag{3-32}$$

$$N = 32 \tag{3-33}$$

$$SCR = 10dB \tag{3-34}$$

$$CNR = 5dB \tag{3-35}$$

$$k = 27 \quad \text{(for OS-CFAR)} \tag{3-36}$$

and the number of trials as

$$n = 1000/P_f = 1{,}000{,}000$$

3.2.2 Simulation Results

In all performance plots in this section, the same legend was used for a given CFAR processor. These legends are as follows:

legend	CFAR Processor
"square"	OS(27)-CFAR
"star"	CA-CFAR
"plus"	GO-CFAR
"diamond"	OS(32)-CFAR(used only in non homogeneous clutter cases)

where 27 and 32 are the order numbers used for the OS-CFAR. In P_f performance plots, the horizontal straight line without a legend on it shows the design P_f value for a Gaussian environment. Eight shape parameter values were selected to run the simulations. These values are 0.10, 0.25, 0.50, 1.50, 2.50, 5.00, 10.00, 50.00 . The smaller values of the shape parameter correspond to spiky clutter and the larger values correspond to nearly Gaussian clutter.

3.2.2.1 Homogeneous Background

In Figures 3-3 through 3-6, the P_f and P_d performance versus shape parameter of K-distributed clutter are shown for K-distributed clutter-only and K-distributed clutter-plus-Gaussian noise homogeneous background conditions. SCR was chosen to be 10 dB and CNR was equal to 5 dB.

In Figures 3-7 and 3-8, the 95% simulation error Confidence Intervals of the P_f and P_d performance curves are shown for all CFAR processors.

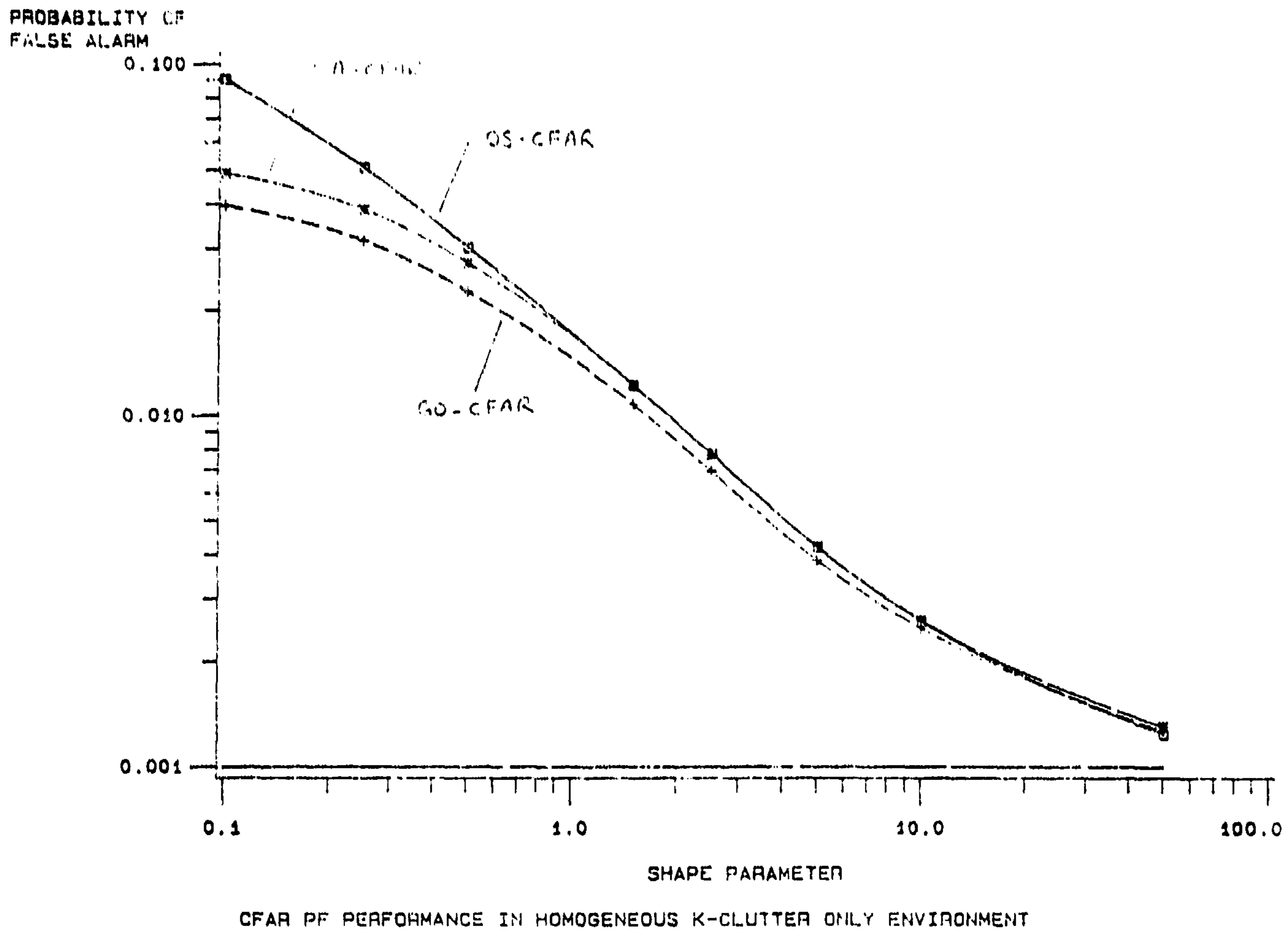


Figure 3-3: Actual probability of false alarm in homogeneous K-distributed clutter as a function of shape parameter. The design P_f is 0.001.

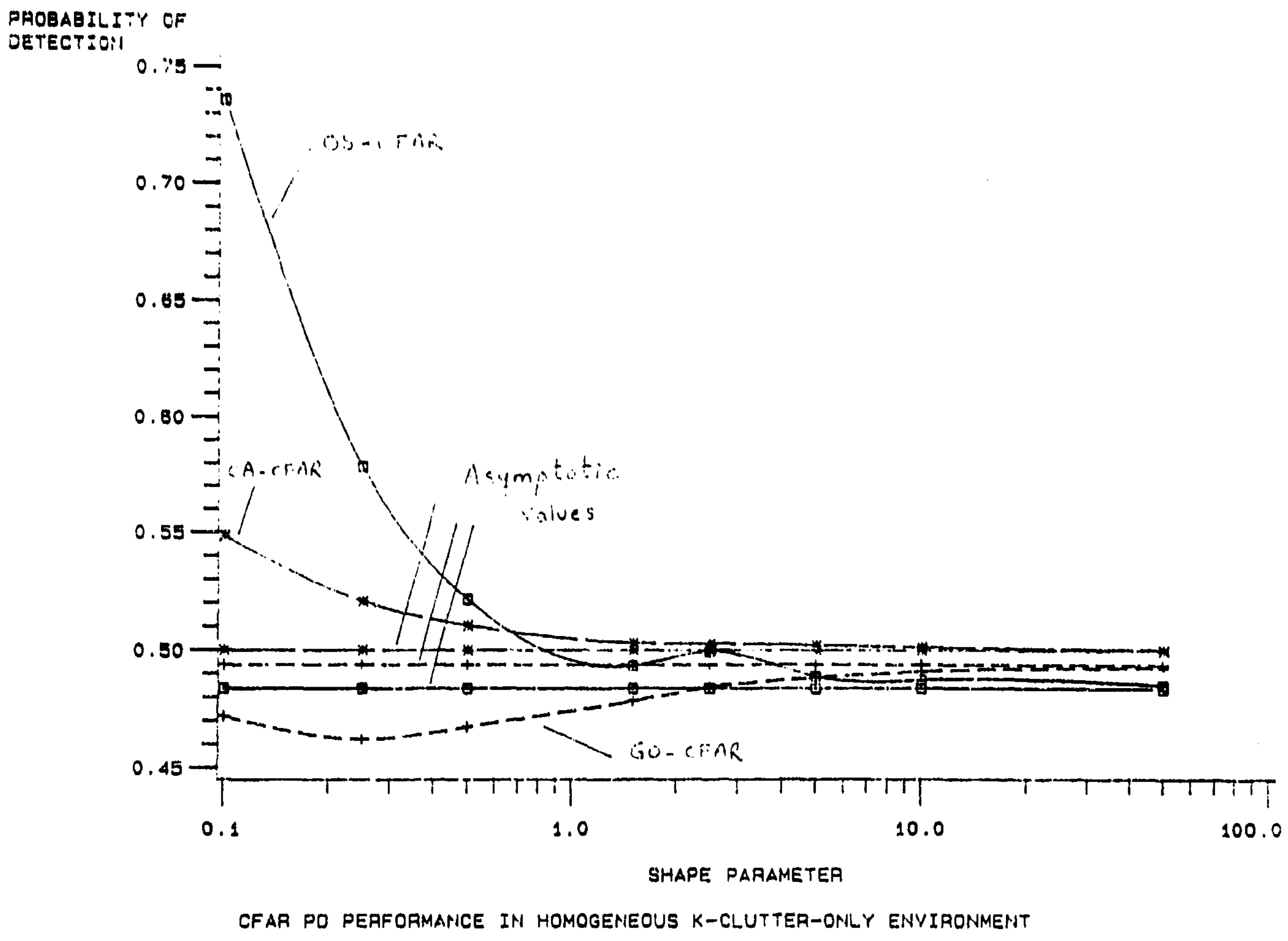


Figure 3-4: Probability of detection in homogeneous K-distributed clutter as a function of shape parameter.

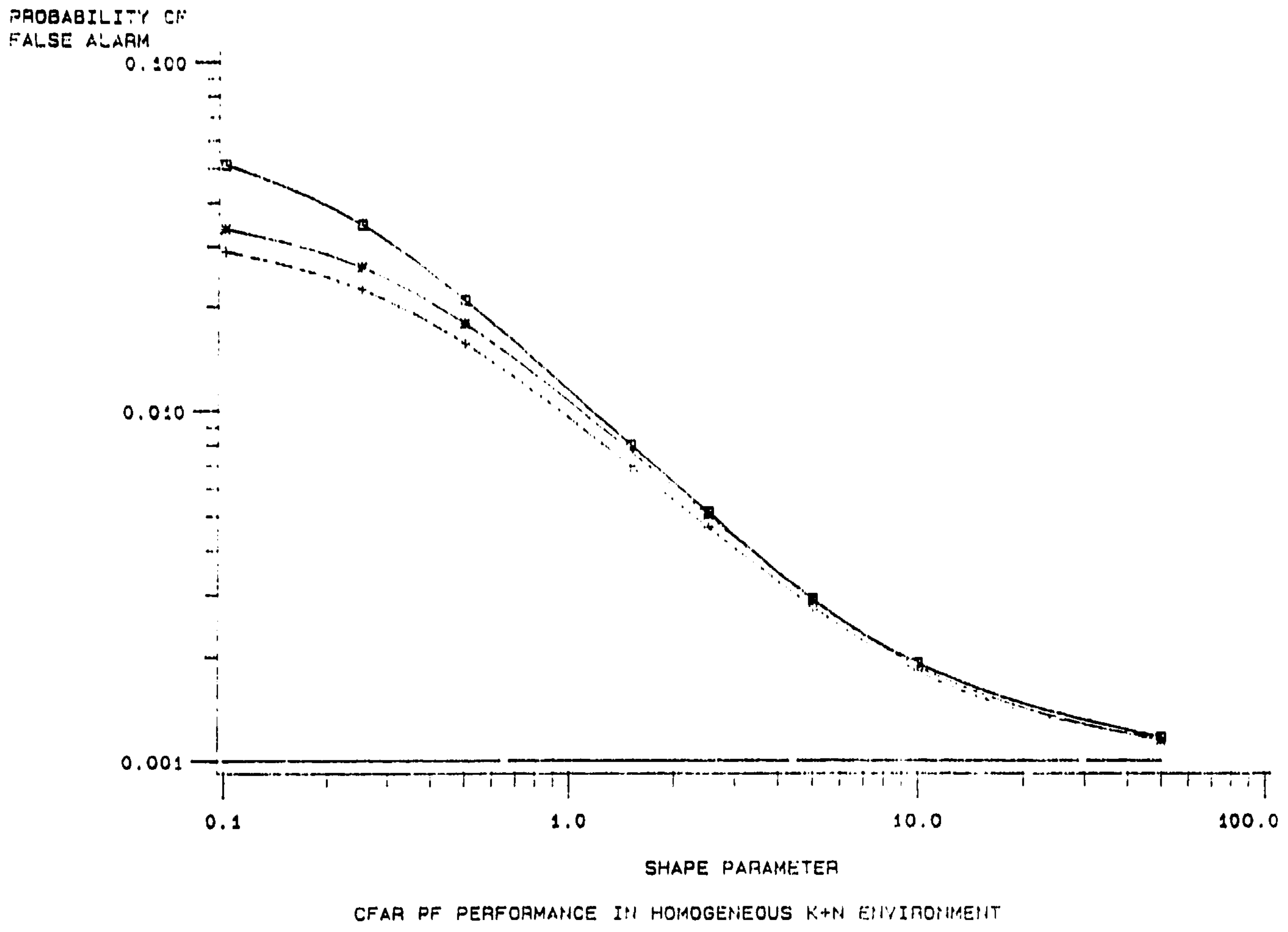


Figure 3-5: Actual probability of false alarm in homogeneous K-distributed clutter plus Gaussian thermal noise as a function of shape parameter. The design Pf is 0.001.

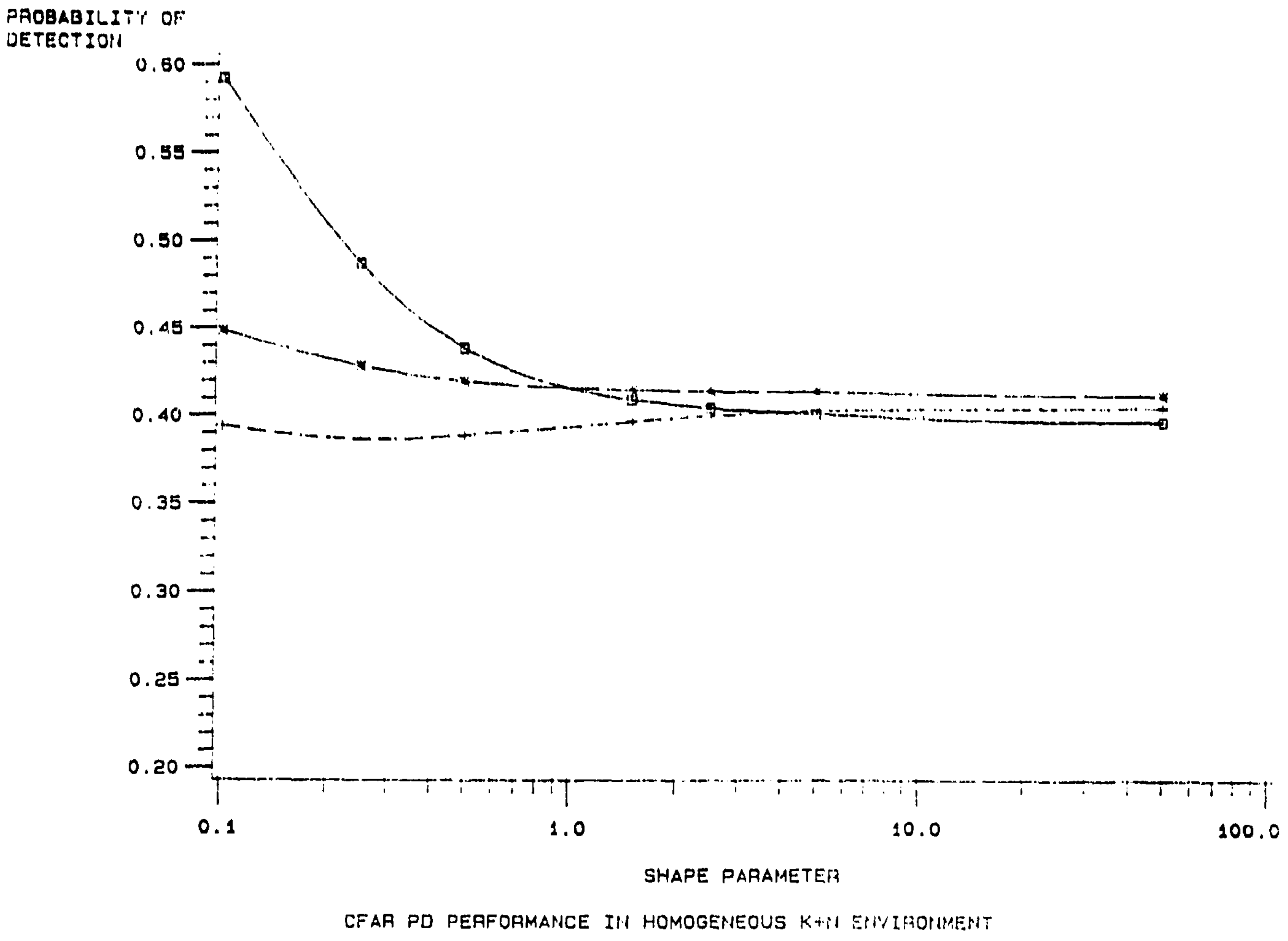


Figure 3-6: Probability of detection in homogeneous K-distributed clutter plus Gaussian thermal noise as a function of shape parameter.

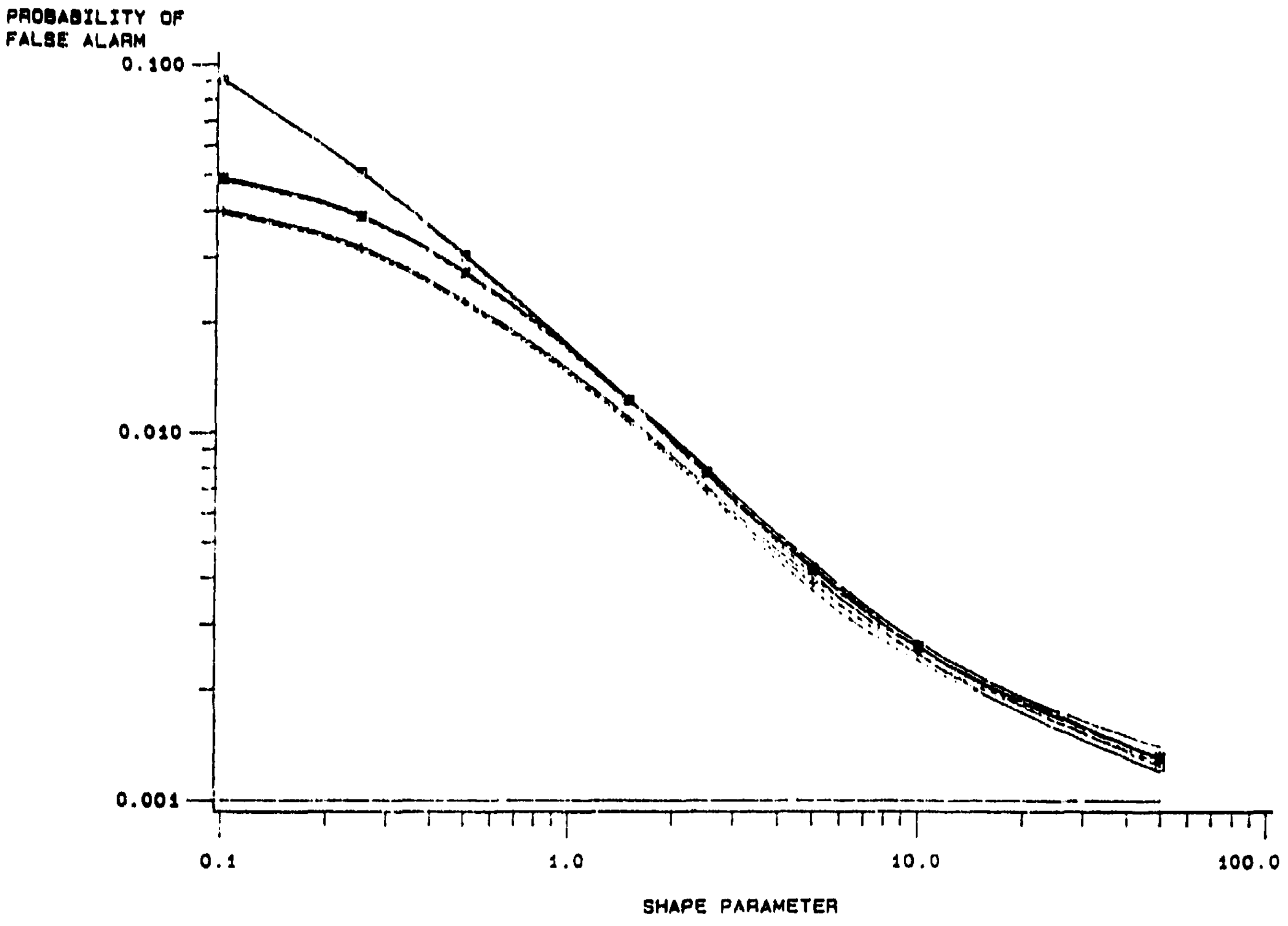


Figure 3-7: The 95% simulation error confidence intervals of P_f.

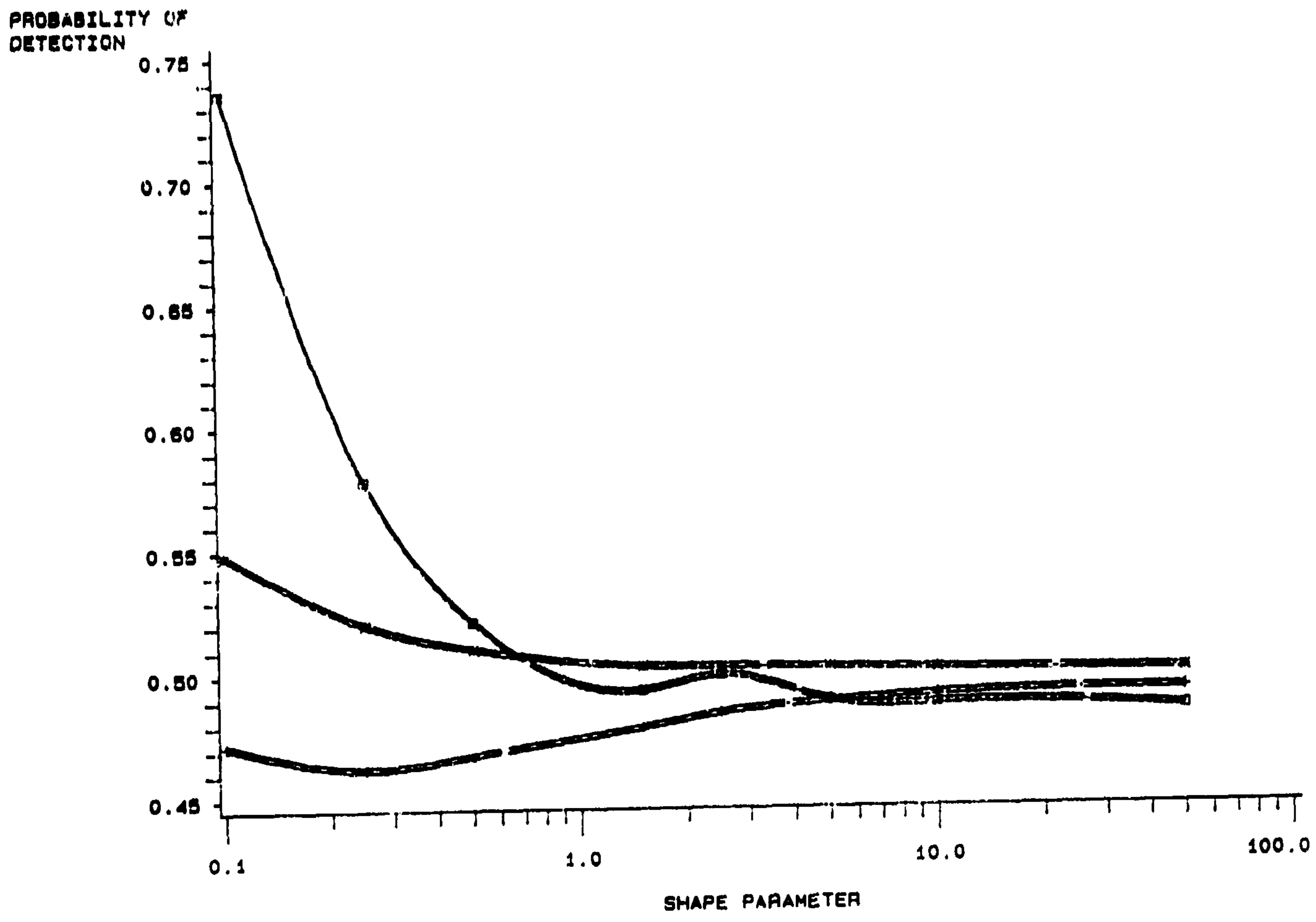


Figure 3-8: The 95% simulation error confidence intervals of P_d.

3.2.2.2 Non homogeneous Background

In this section, two cases of non homogeneous background conditions have been considered: multiple targets and non homogeneous clutter

Multiple Targets

In Figures 3-9 through 3-14, the P_f and P_d performances versus shape parameter α of K-distributed clutter are shown for a K-distributed clutter-only background condition with an embedded interfering target in it. As before, SCR was chosen to be 10 dB and CNR was equal to 5 dB. Three primary to interfering target power levels were considered: SIR = 5 dB, 0 dB and -5 dB.

In Figures 3-15 through 3-20, the same cases were considered as in Figures 3-9 through 3-14, but this time the background was filled with K-distributed clutter-plus-Gaussian noise instead of K-distributed clutter only.

Non homogeneous Clutter

In this section, two different order numbers were used for each case of the OS-CFAR processor. That is k1=27 and k2=32.

In Figures 3-21 through 3-26, the P_f performances are shown versus α. In these cases, while some of the reference window cells were filled with point K-distributed clutter plus Gaussian noise, the rest were filled with only Gaussian noise. This may happen if the clutter starts to occupy the reference window cells one by one from one side. This scenario was assumed to be happening in these simulations which was important for GO-CFAR. For CA-CFAR and OS-CFAR, the location of the clutter within the reference window is not important. The test cell (TC) was assumed to be filled with only Gaussian-noise . CNR was equal to 10 dB. The performances are shown for α values: 0.10, 0.25, 0.50, 1.50, 2.50, 5.00.

In Figures 3-27 through 3-32, the same conditions were repeated as in above, but this time the test cell (TC) was assumed to be filled with K-distributed clutter-plus-Gaussian noise.

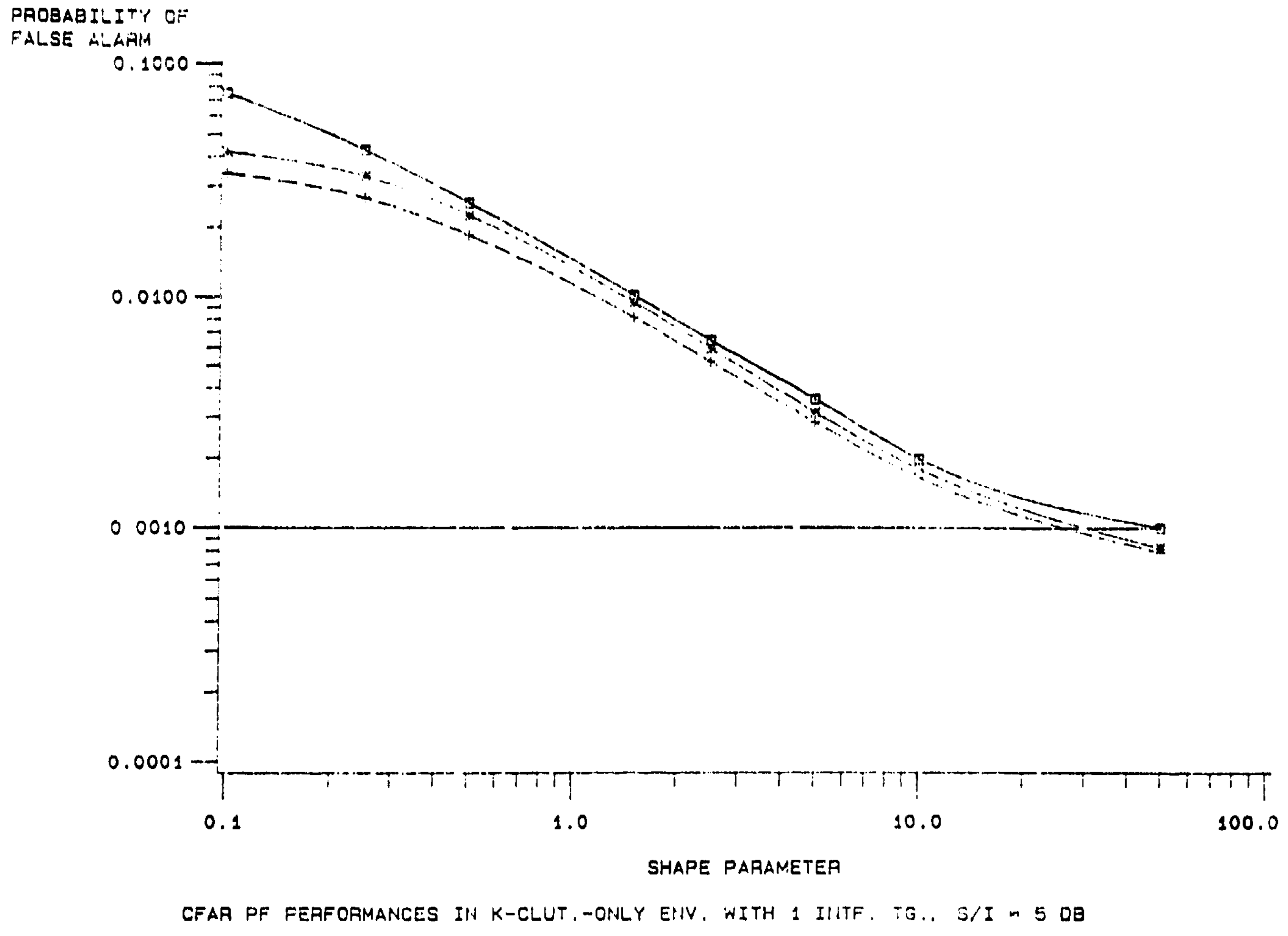


Figure 3-9: Actual probability of false alarm in K-distributed clutter with one interfering target (S/S_I = 5 dB). Design P_f is 0.001.

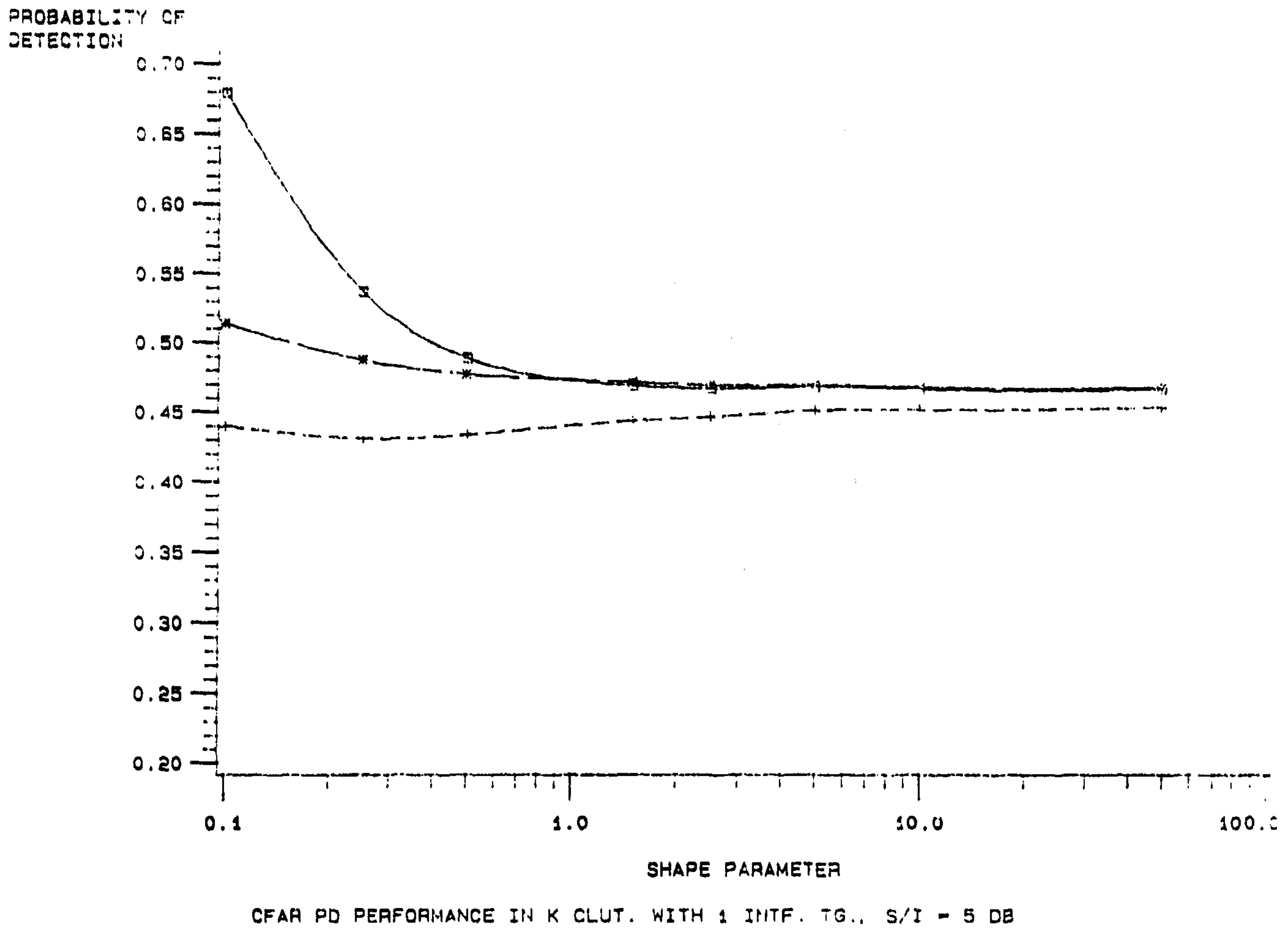


Figure 3-10: Probability of detection in K-distributed clutter with one interfering target(S/S_I = 5 dB)

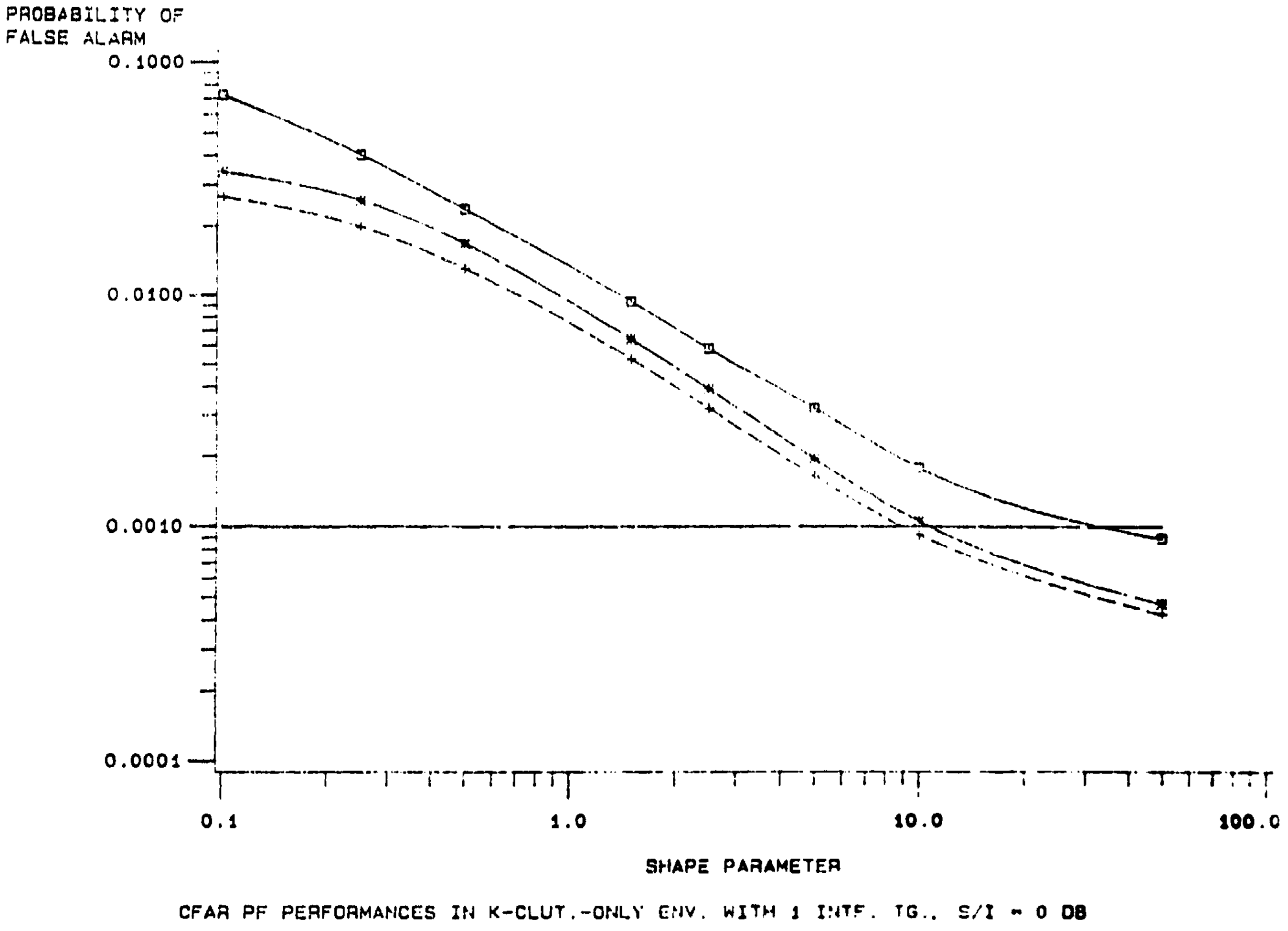


Figure 3-11: Actual probability of false alarm in K-distributed clutter with one interfering target (S/S_I = 0 dB). Design P_f is 0.001.

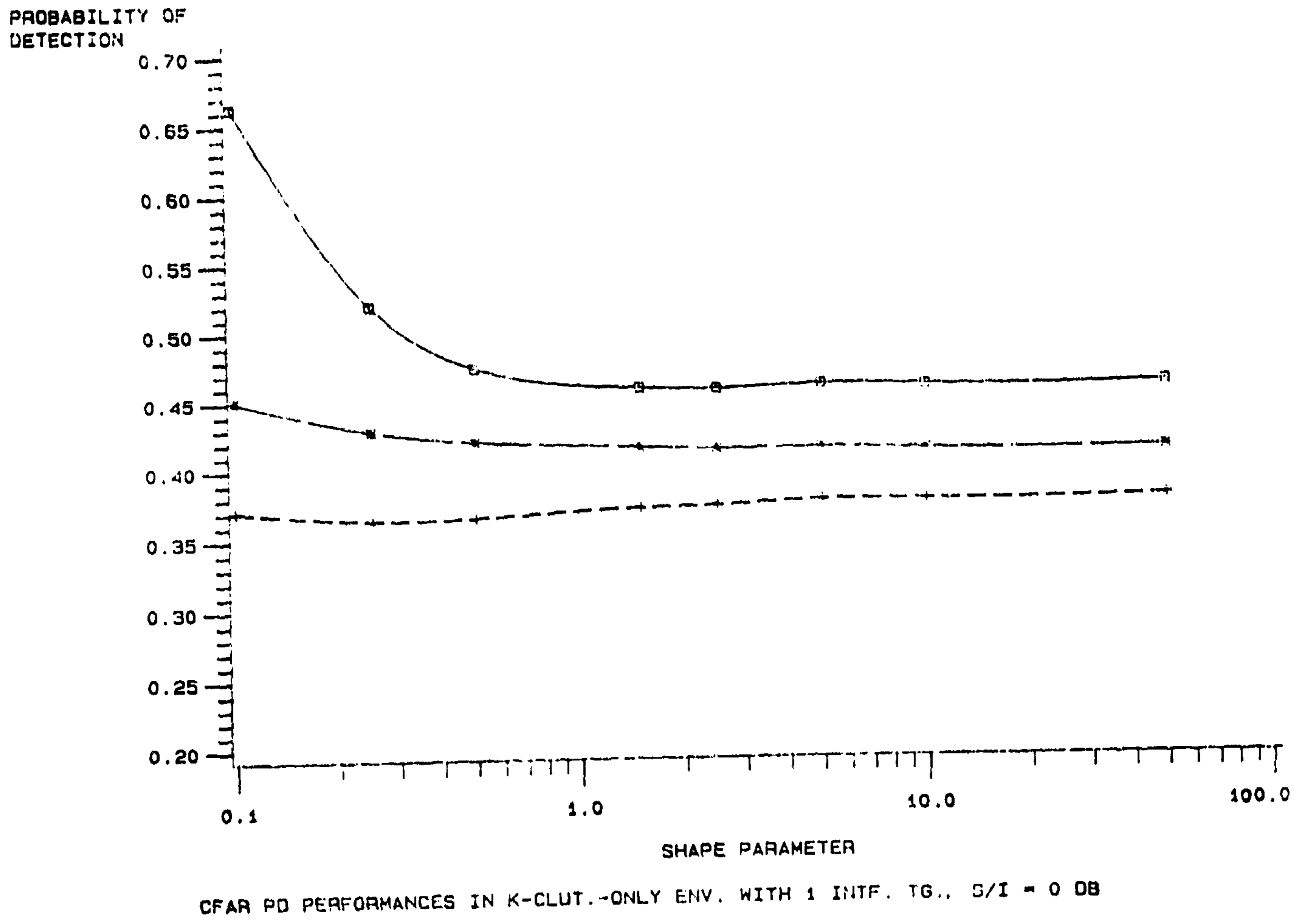


Figure 3-12: Probability of detection in K-distributed clutter with one interfering target(S/S_I = 0 dB)

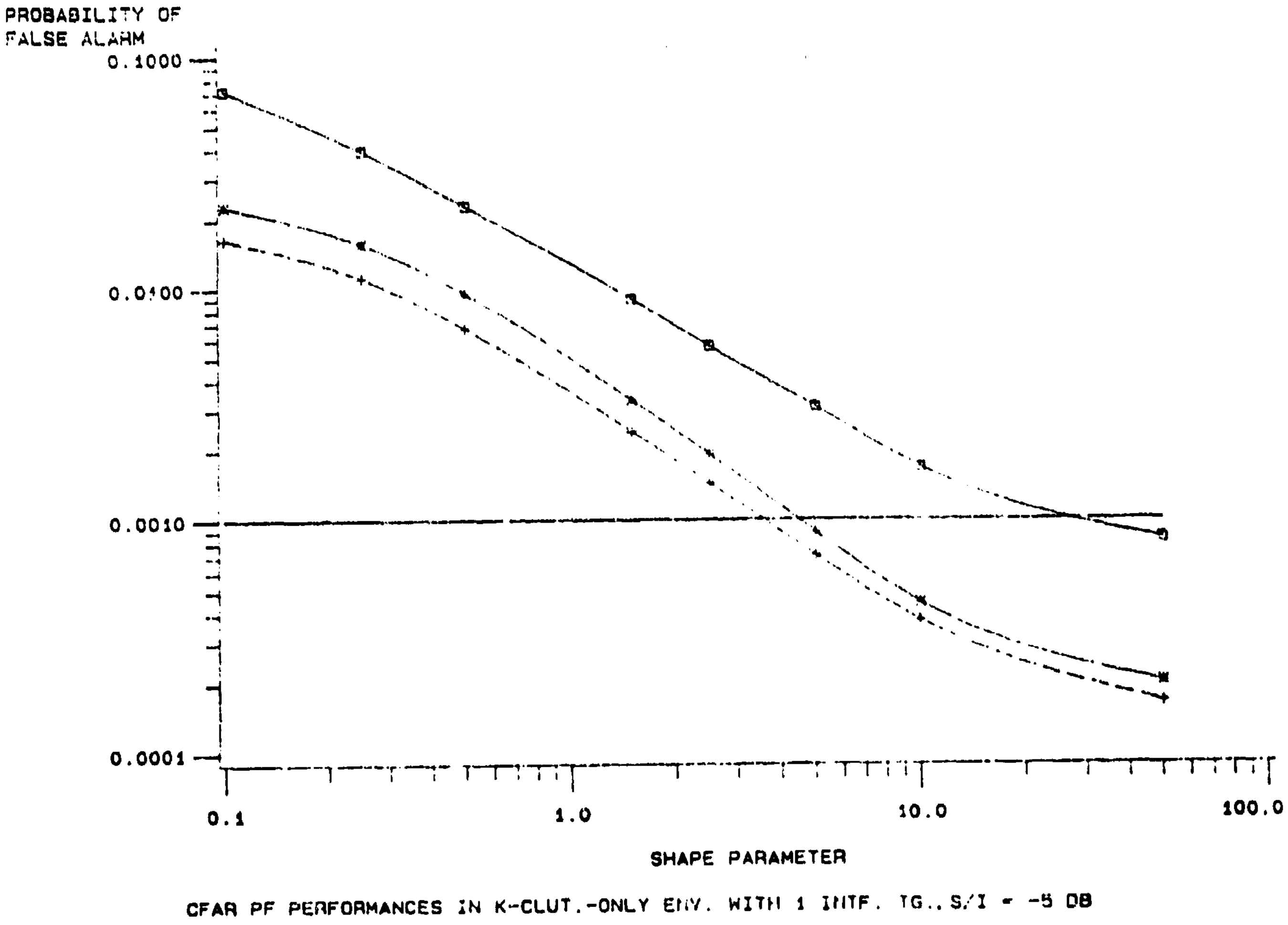


Figure 3-13: Actual probability of false alarm in K-distributed clutter with one interfering target (S/S_I = -5 dB). Design P_f is 0.001.

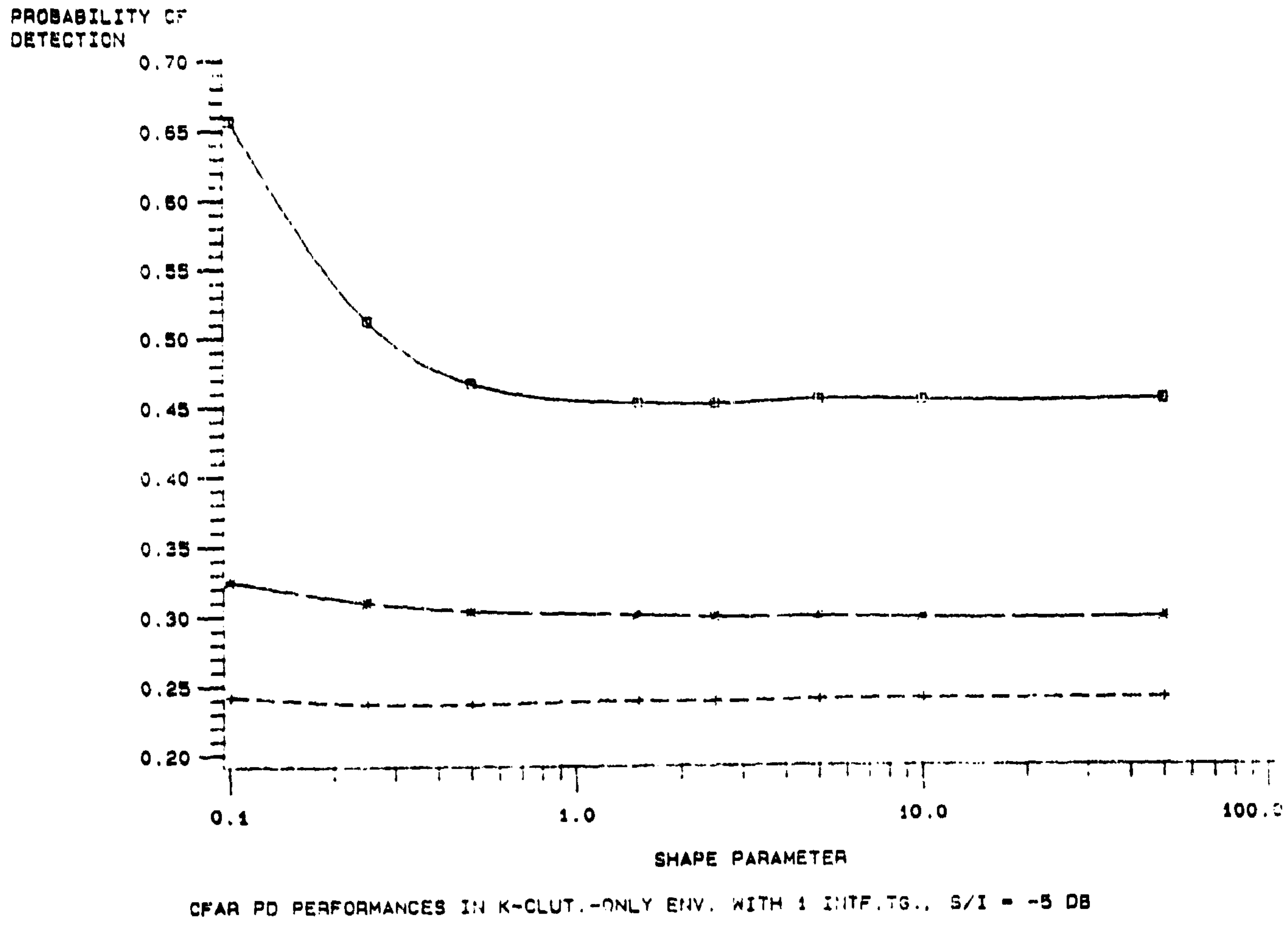


Figure 3-14: Probability of detection in K-distributed clutter with one interfering target(S/S_I = -5 dB)

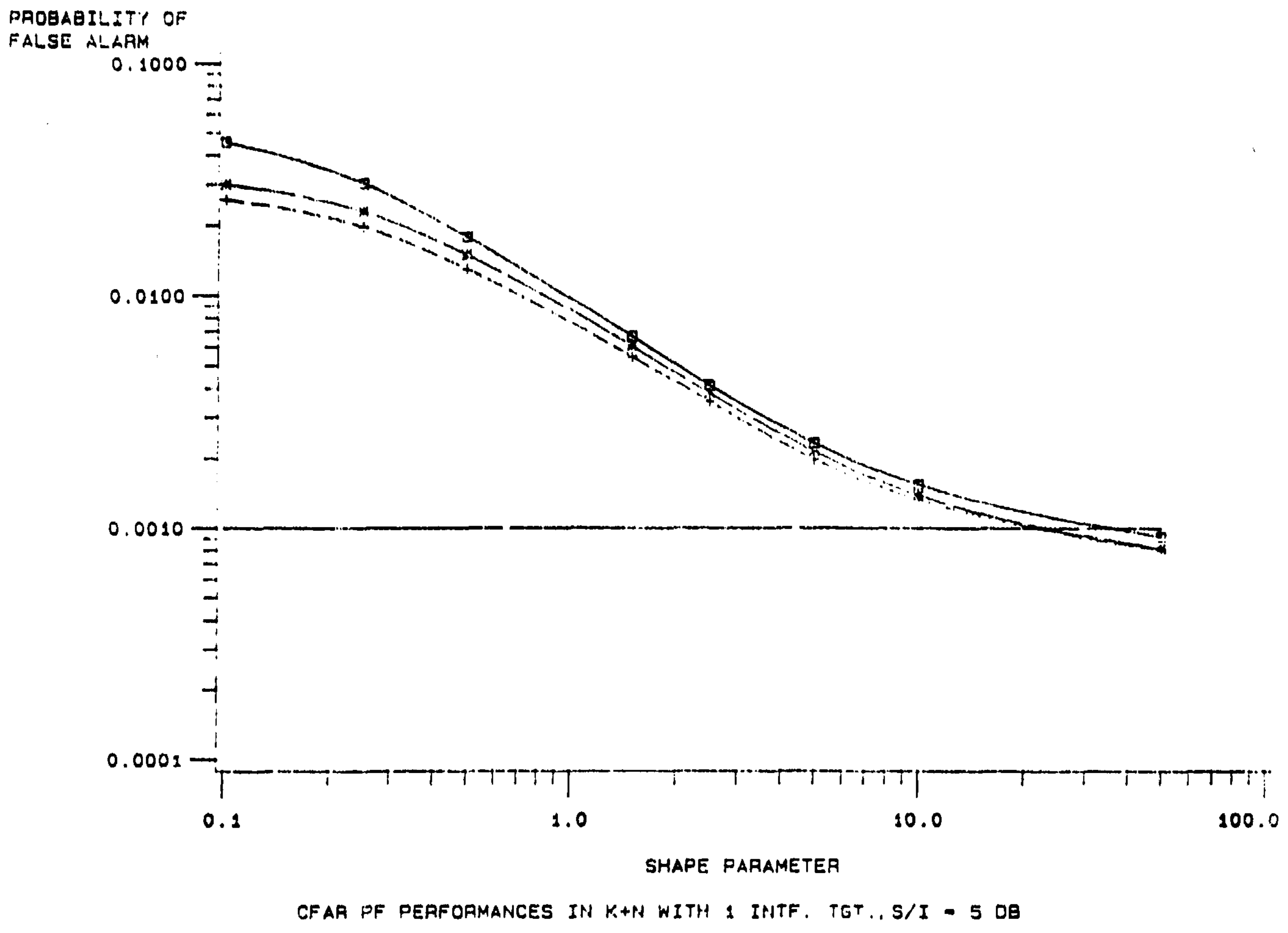


Figure 3-15: Actual probability of false alarm in K-distributed clutter plus Gaussian noise with one interfering target ($S/S_I = 5$ dB).

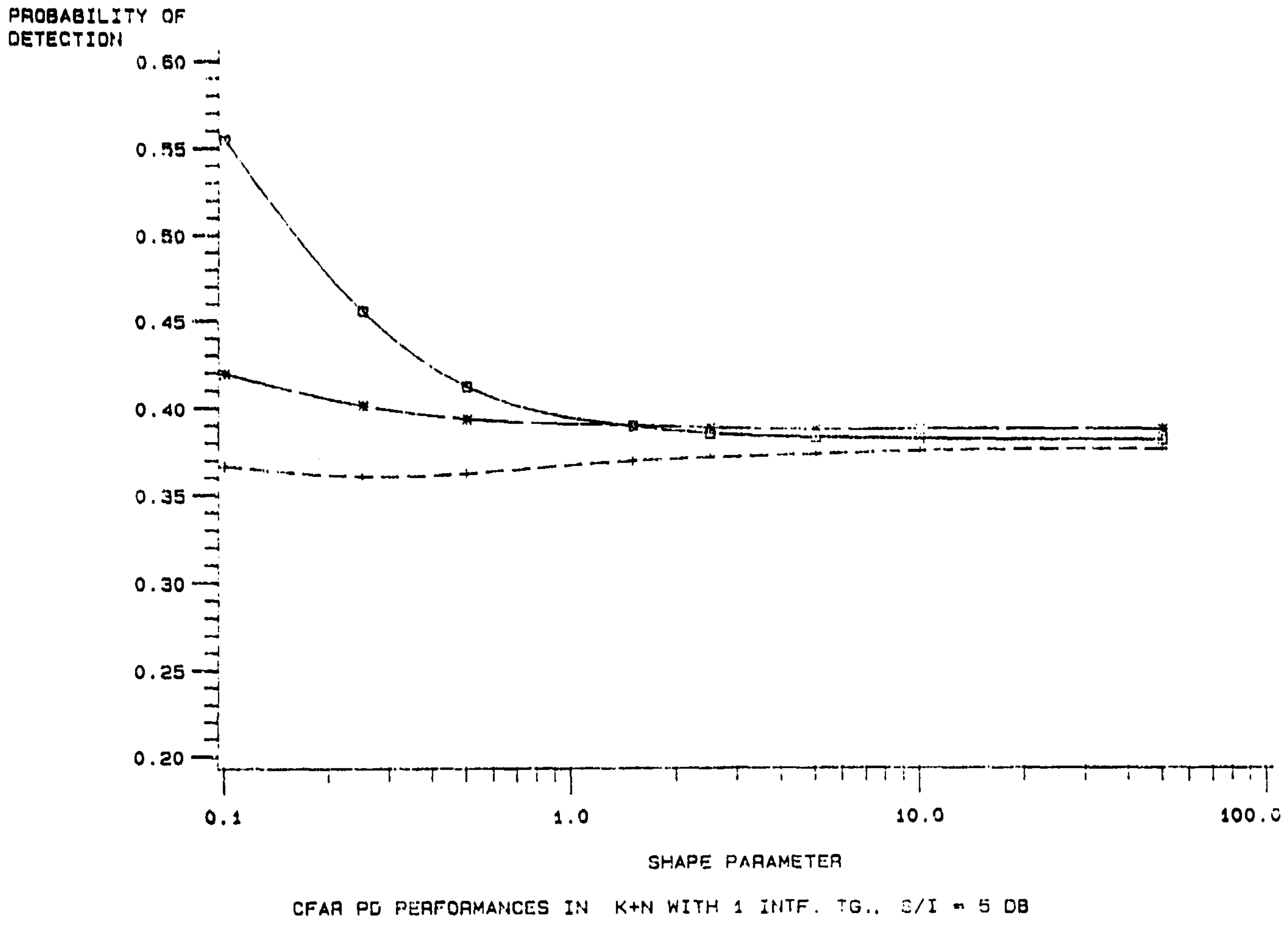


Figure 3-16: Probability of detection in K-distributed clutter plus Gaussian noise with one interfering target (S/S_I = 5 dB).

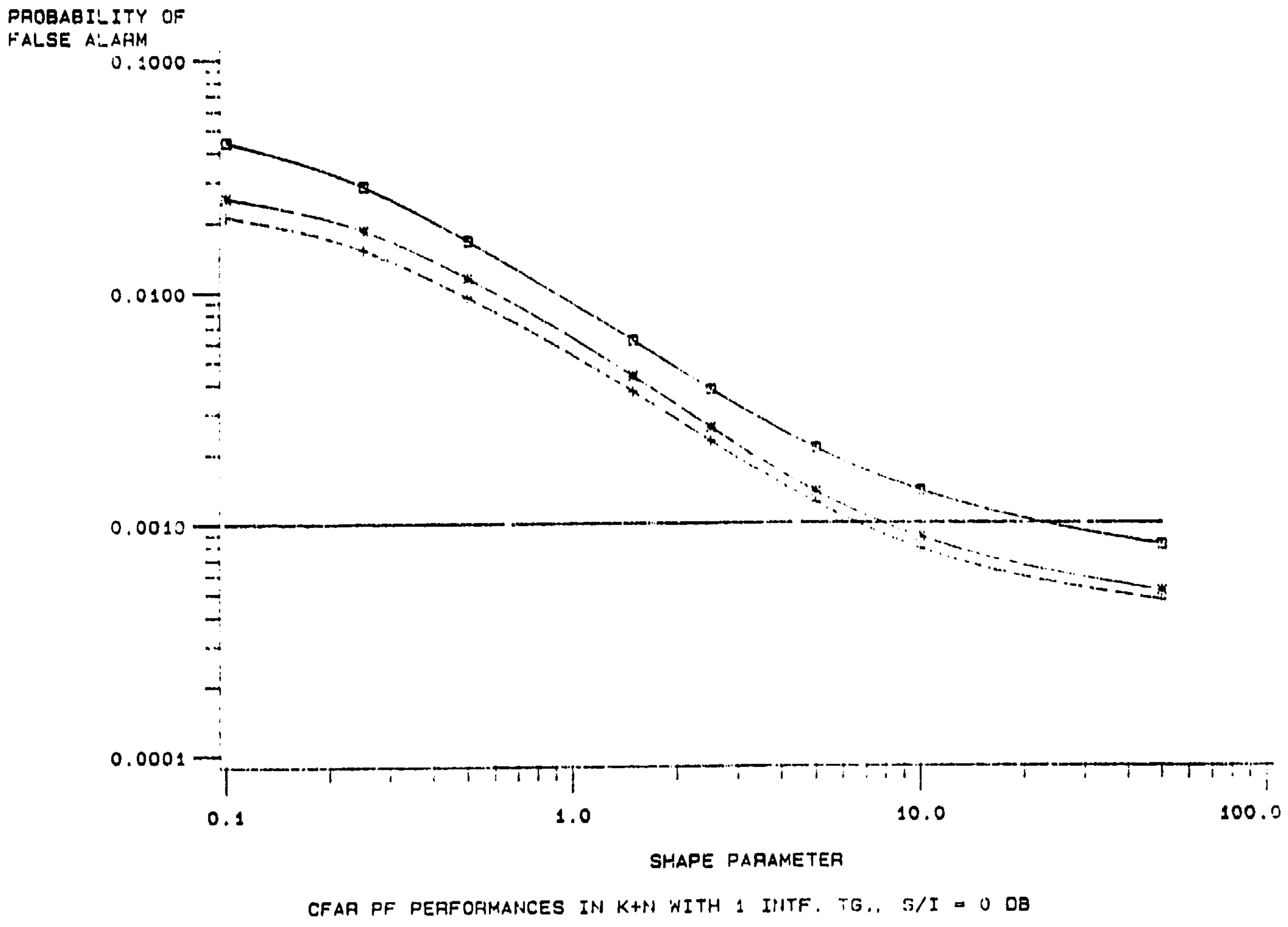


Figure 3-17: Actual probability of false alarm in K-distributed clutter plus Gaussian noise with one interfering target ($S/S_I = 0$ dB).

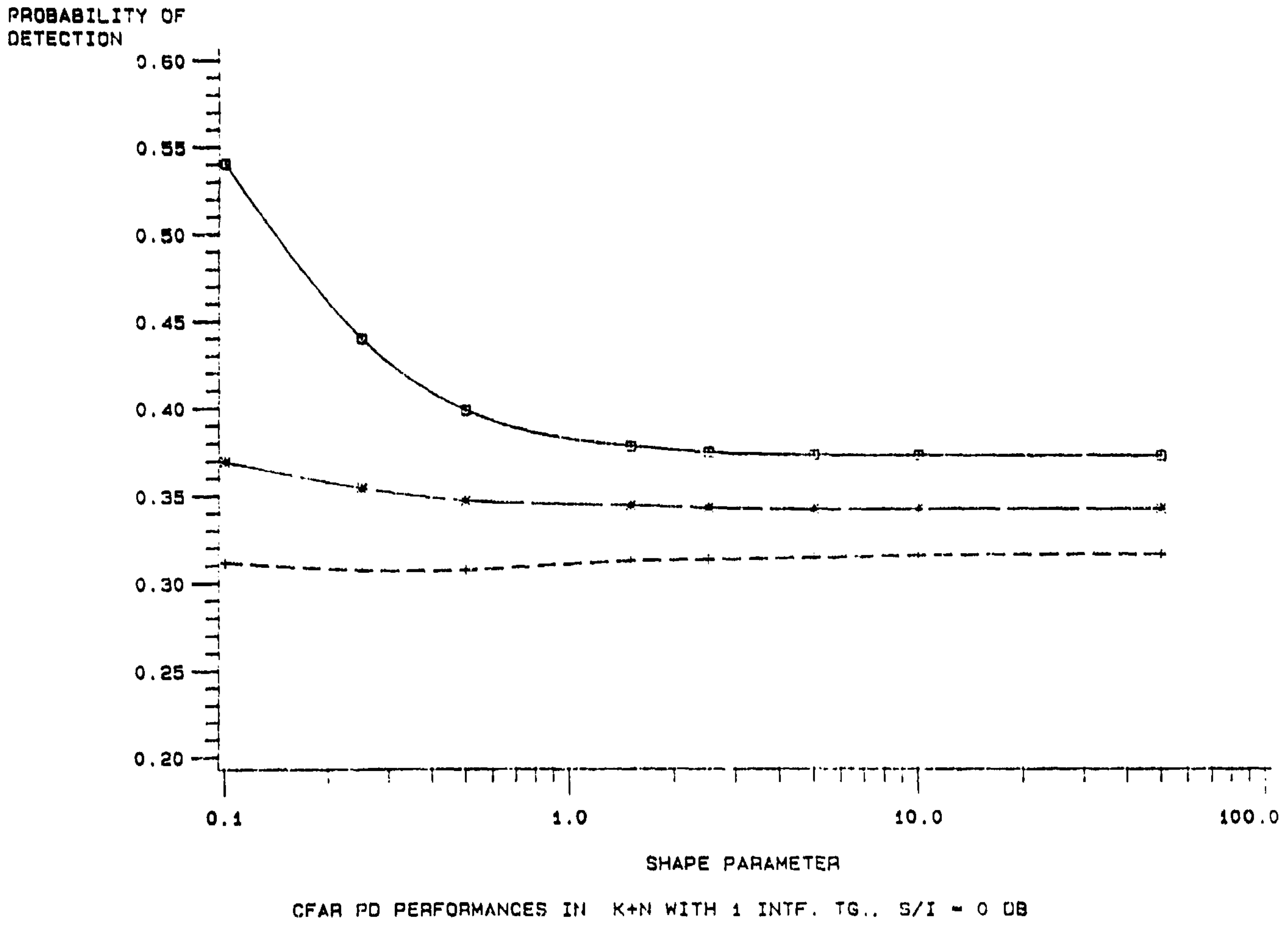


Figure 3-18: Probability of detection in K-distributed clutter plus Gaussian noise with one interfering target ($S/S_I = 0$ dB).

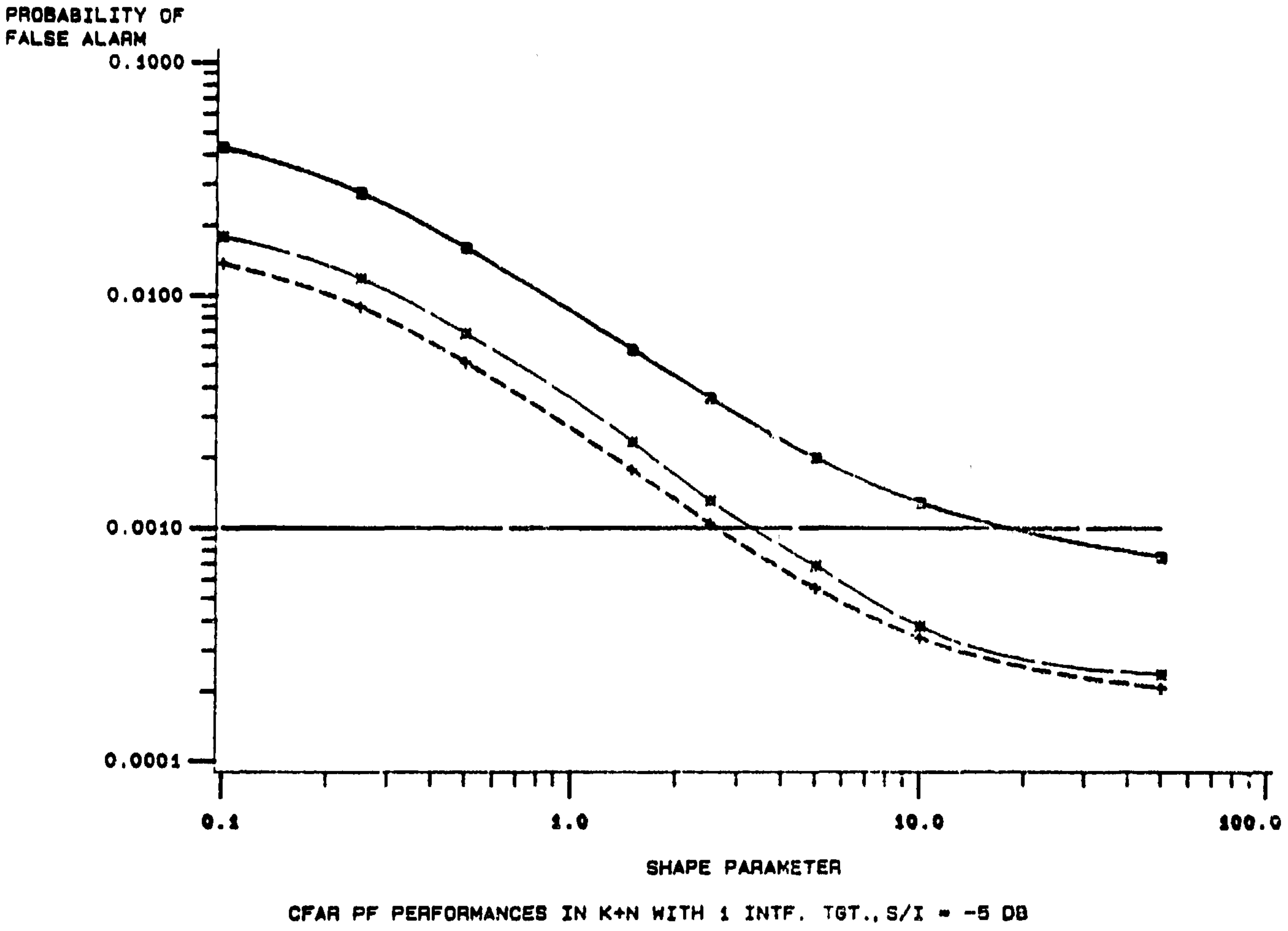


Figure 3-19: Actual probability of false alarm in K-distributed clutter plus Gaussian noise with one interfering target (S/S_I = -5 dB).

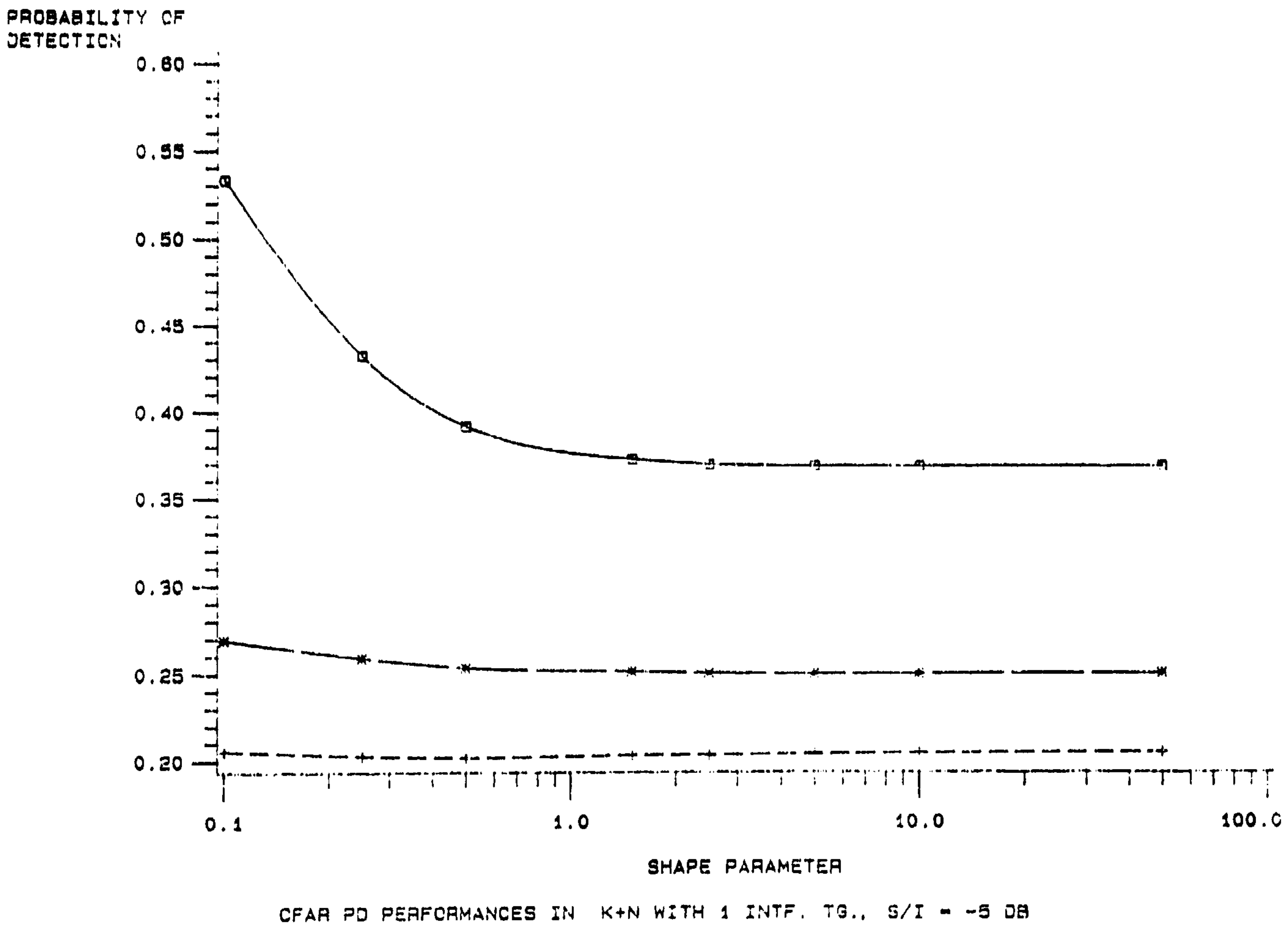


Figure 3-20: Probability of detection in K-distributed clutter plus Gaussian noise with one interfering target (S/S_I = -5 dB).

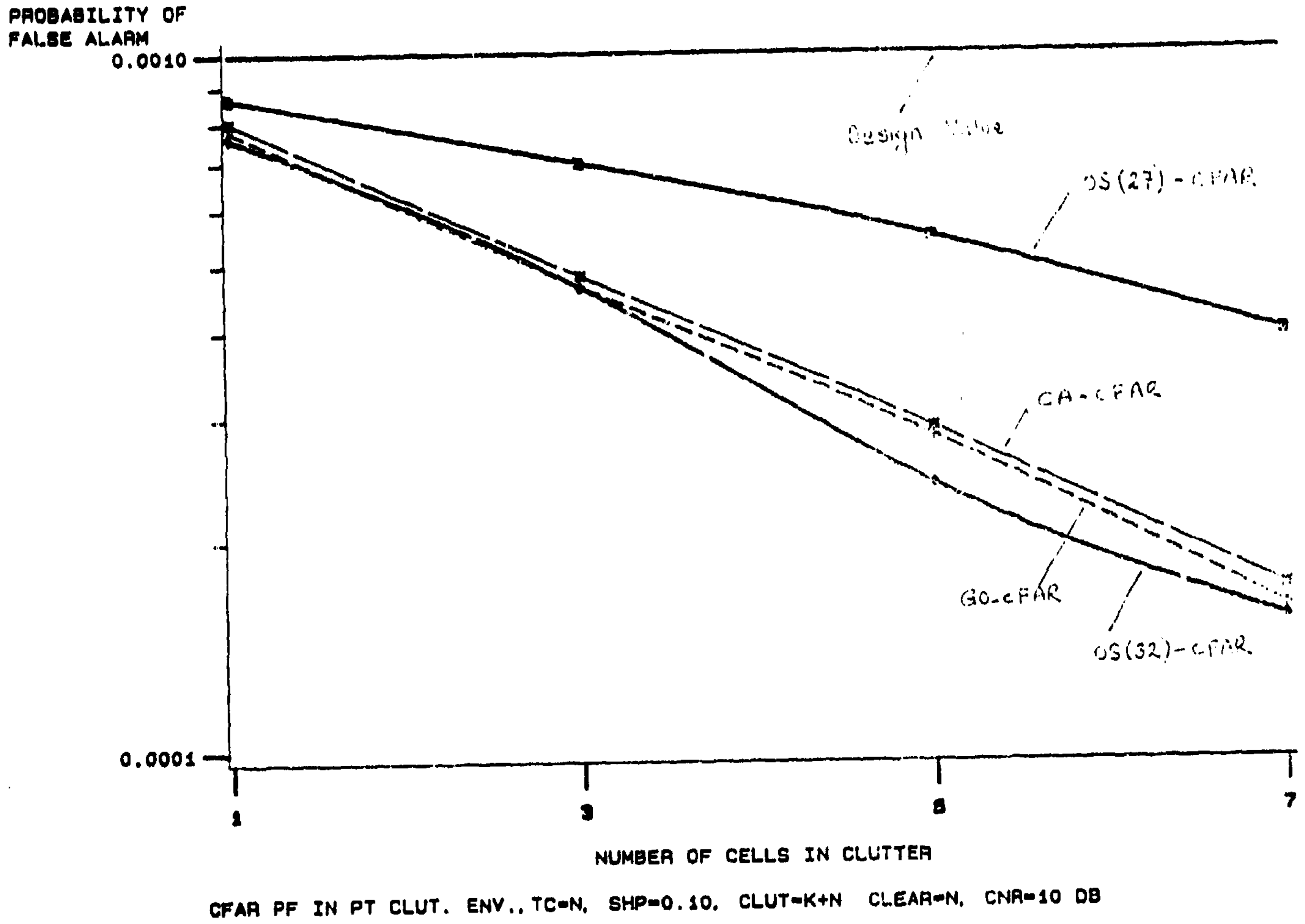


Figure 3-21: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.10. CNR is 10 dB. Test cell is in Gaussian noise only.

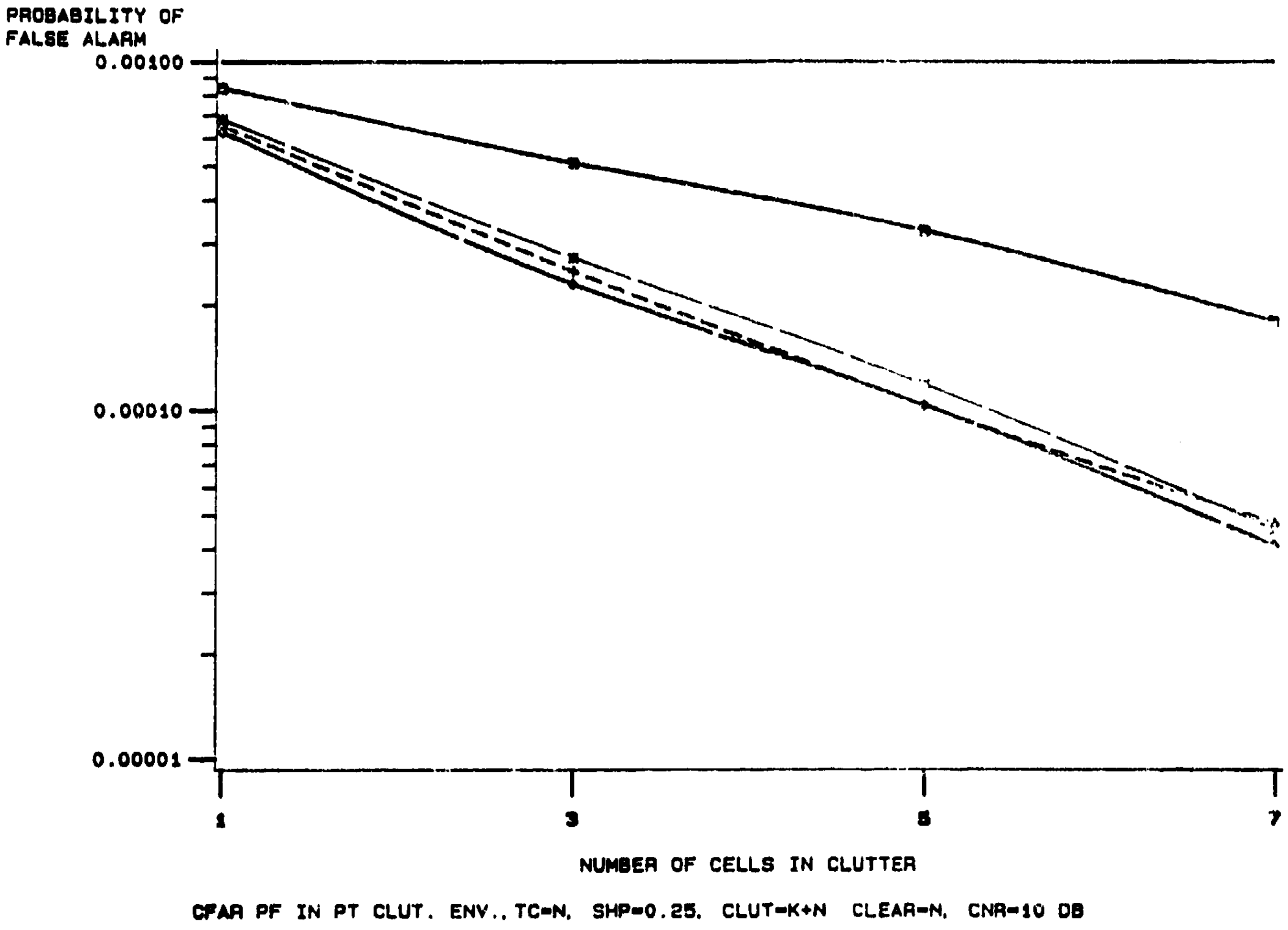


Figure 3-22: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.25. CNR is 10 dB. Test cell is in Gaussian noise only.

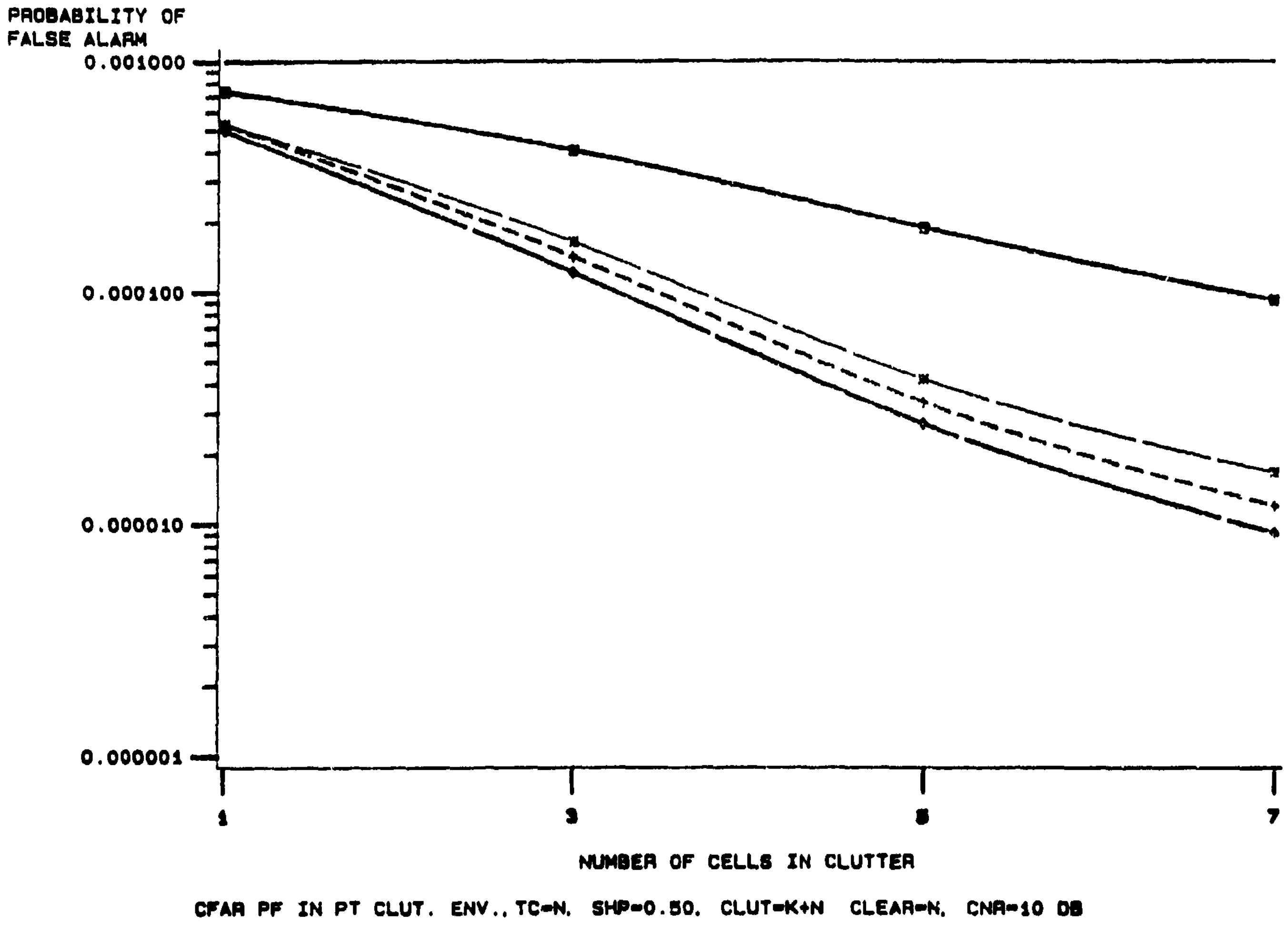


Figure 3-23: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.50. CNR is 10 dB. Test cell is in Gaussian noise only.

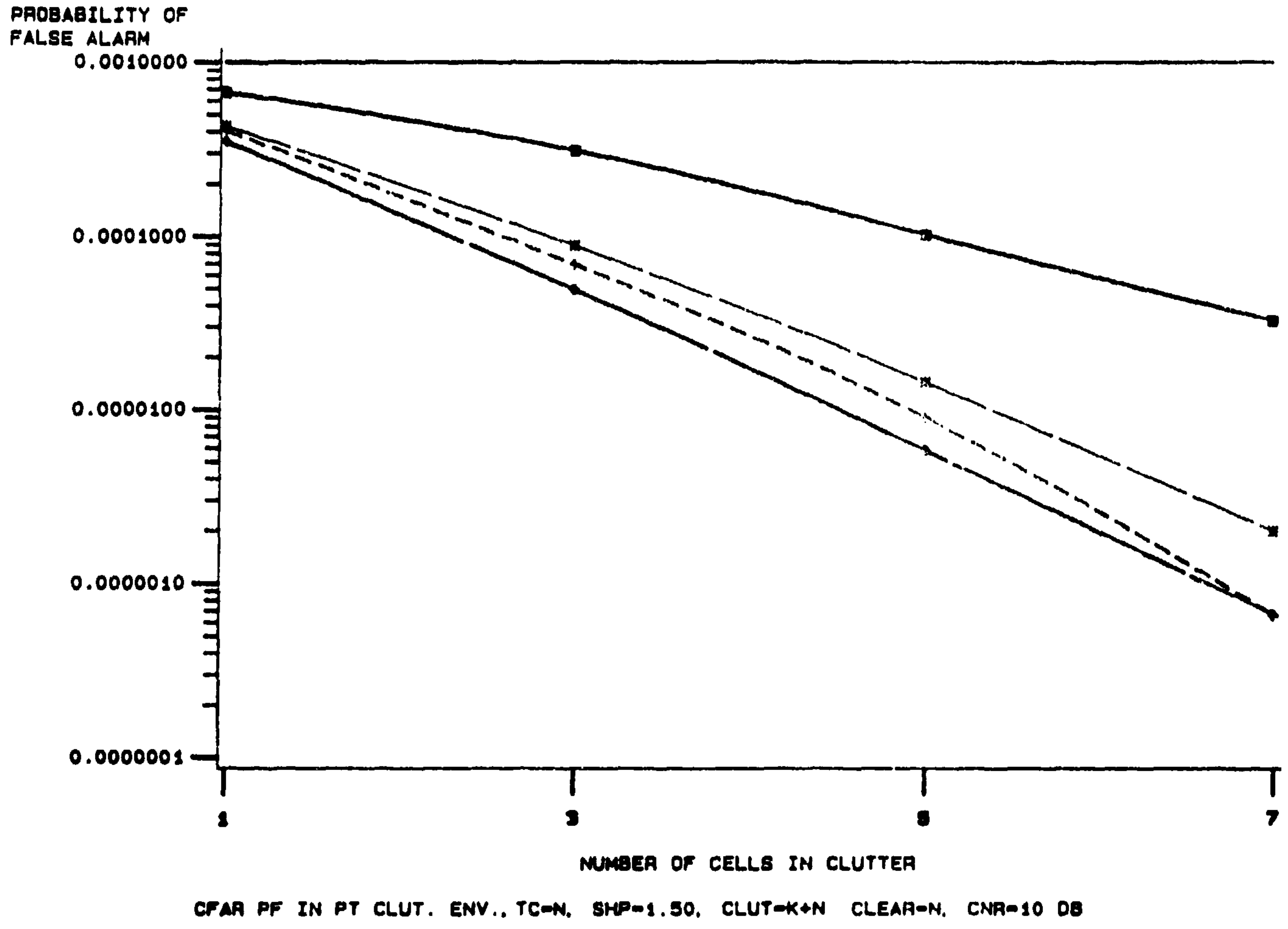


Figure 3-24: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 1.50. CNR is 10 dB. Test cell is in Gaussian noise only.

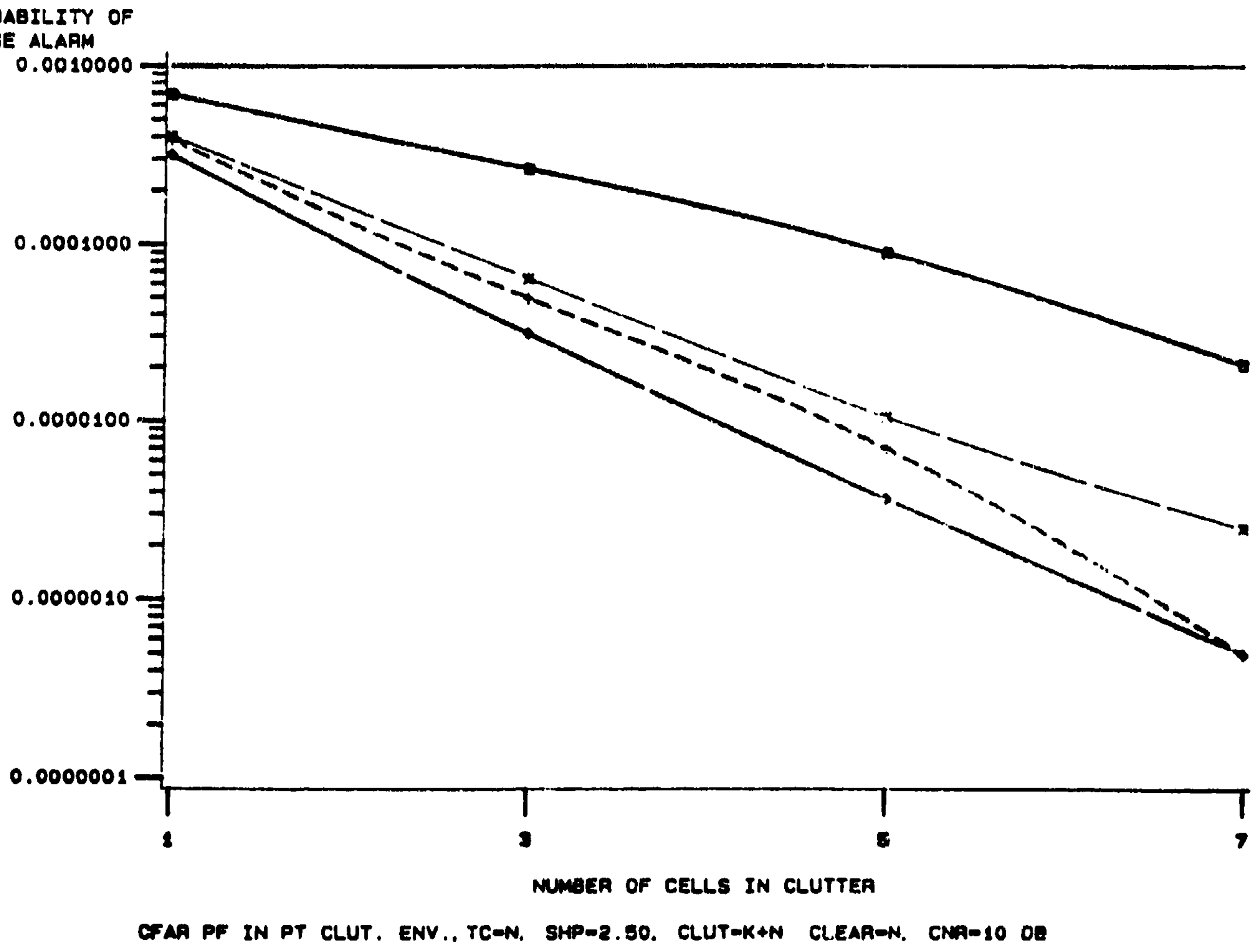


Figure 3-25: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 2.50. CNR is 10 dB. Test cell is in Gaussian noise only.

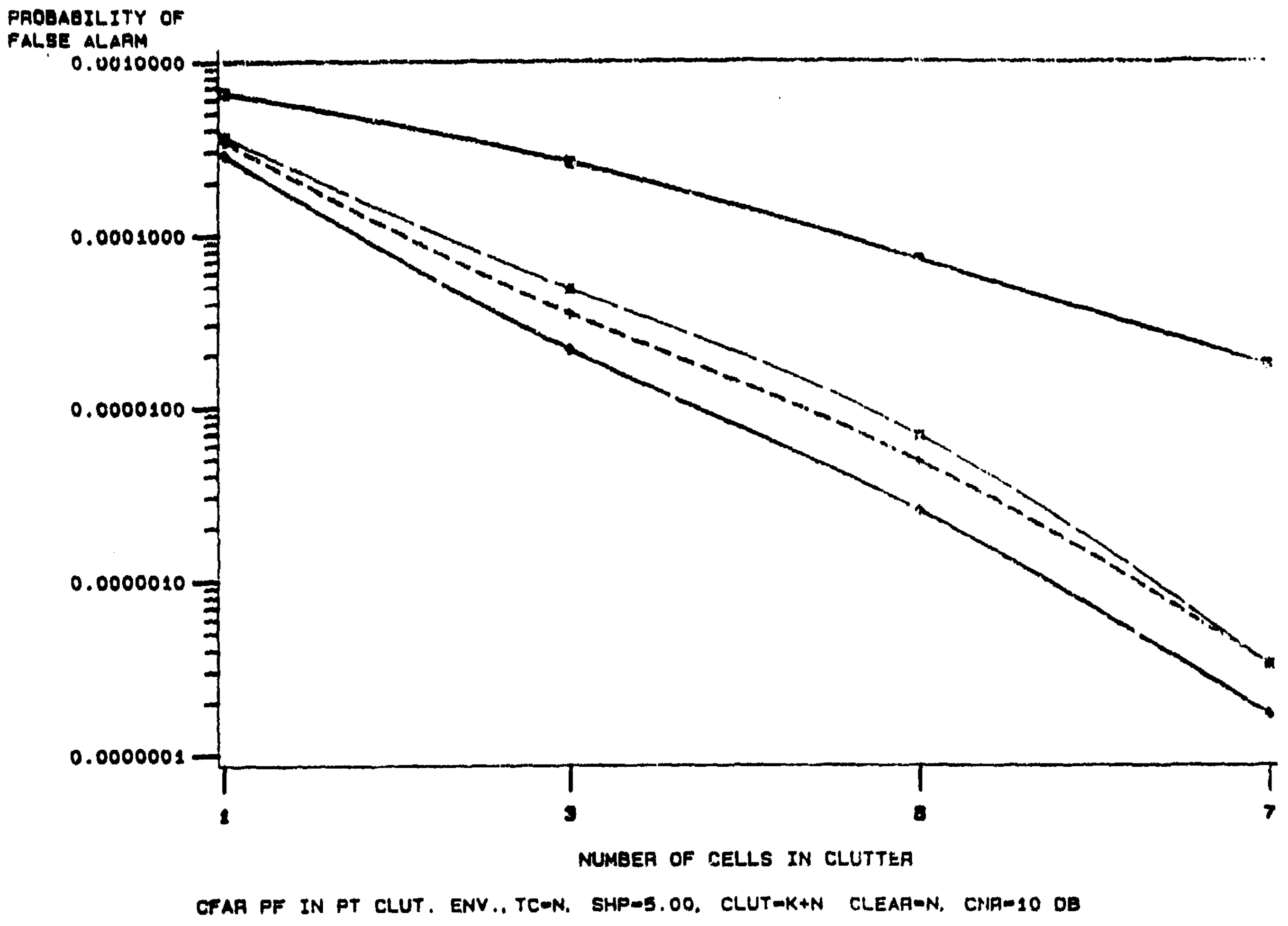


Figure 3-26: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 5.00. CNR is 10 dB. Test cell is in Gaussian noise only.

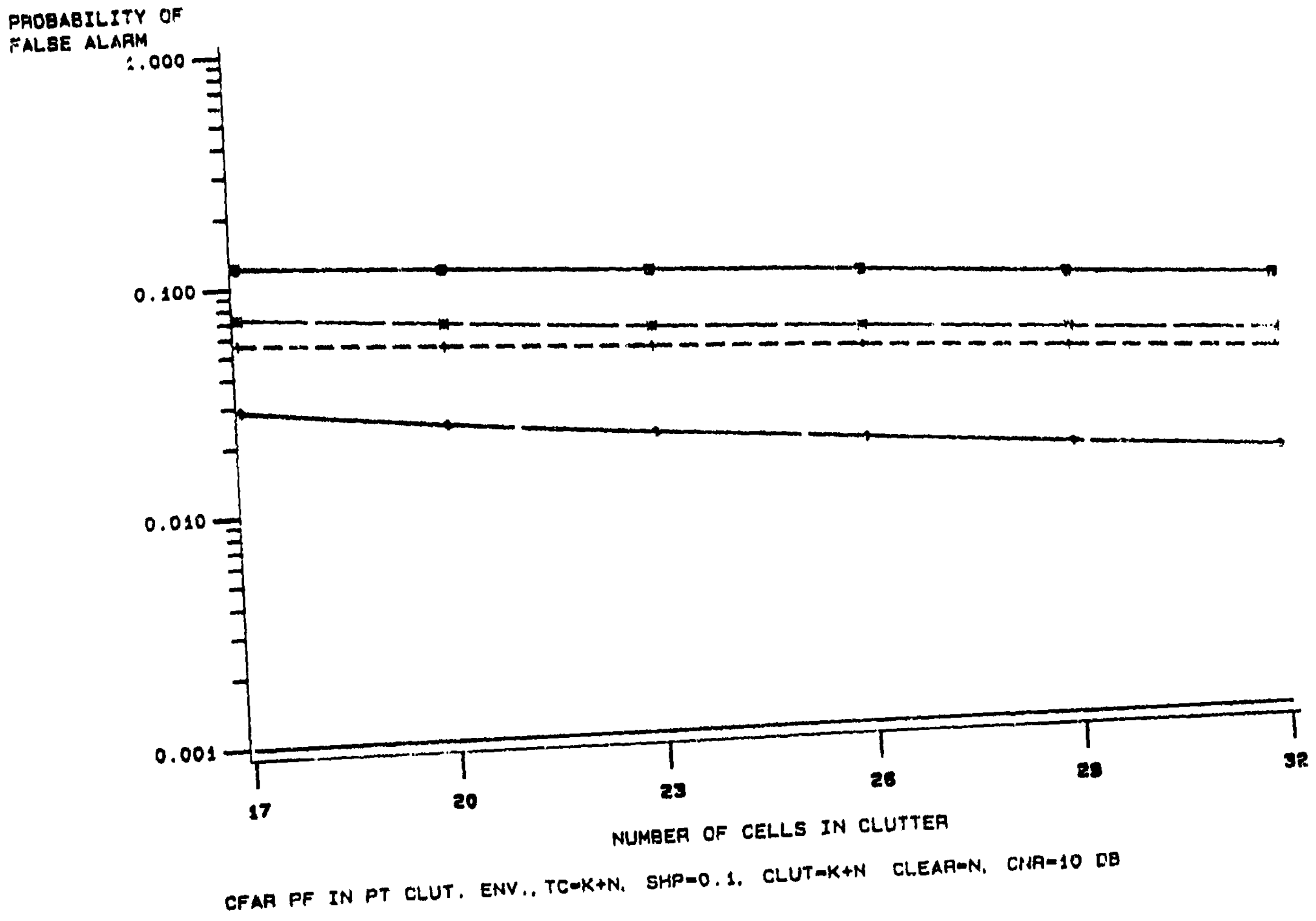


Figure 3-27: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.10. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

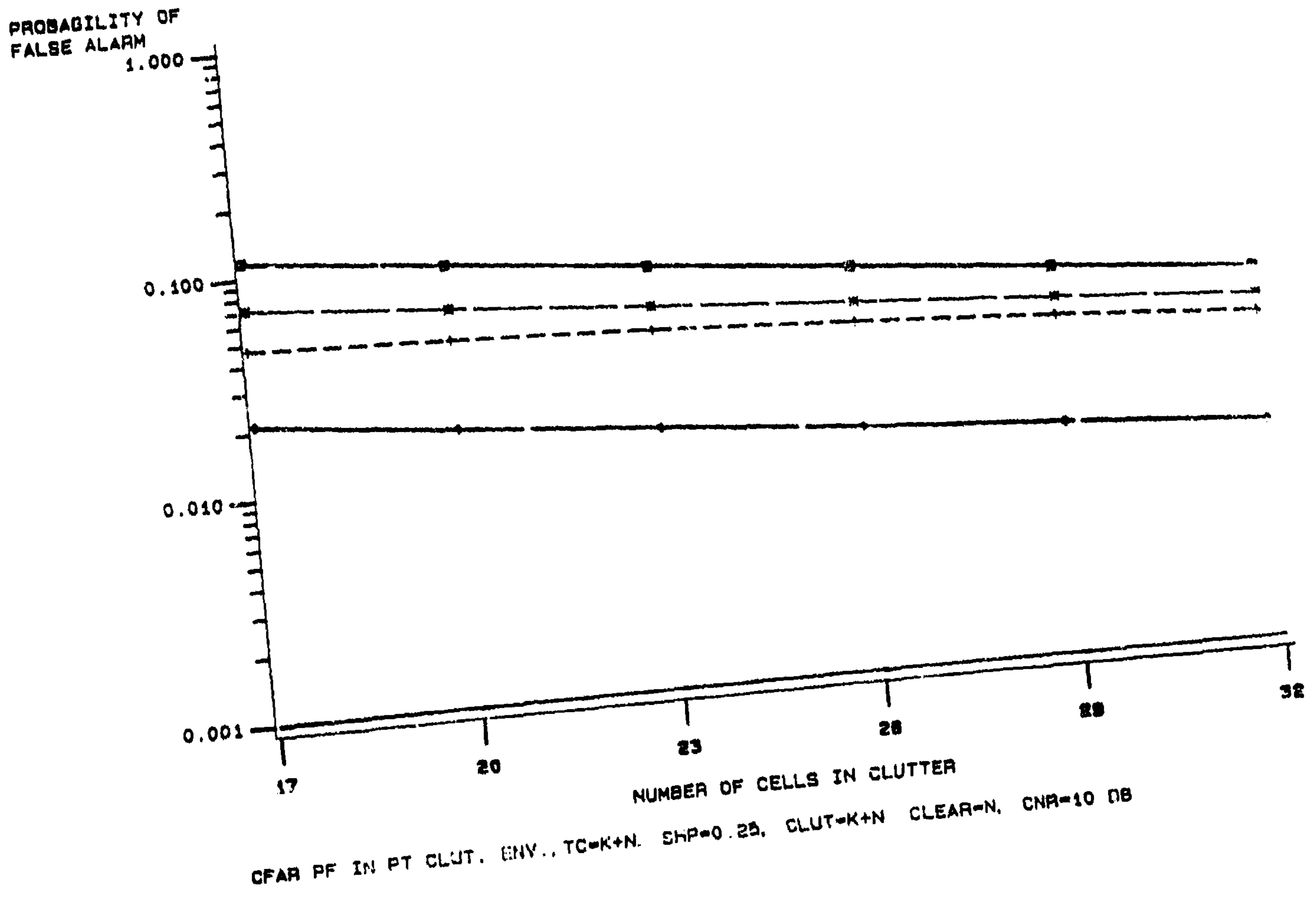


Figure 3-28: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.25. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

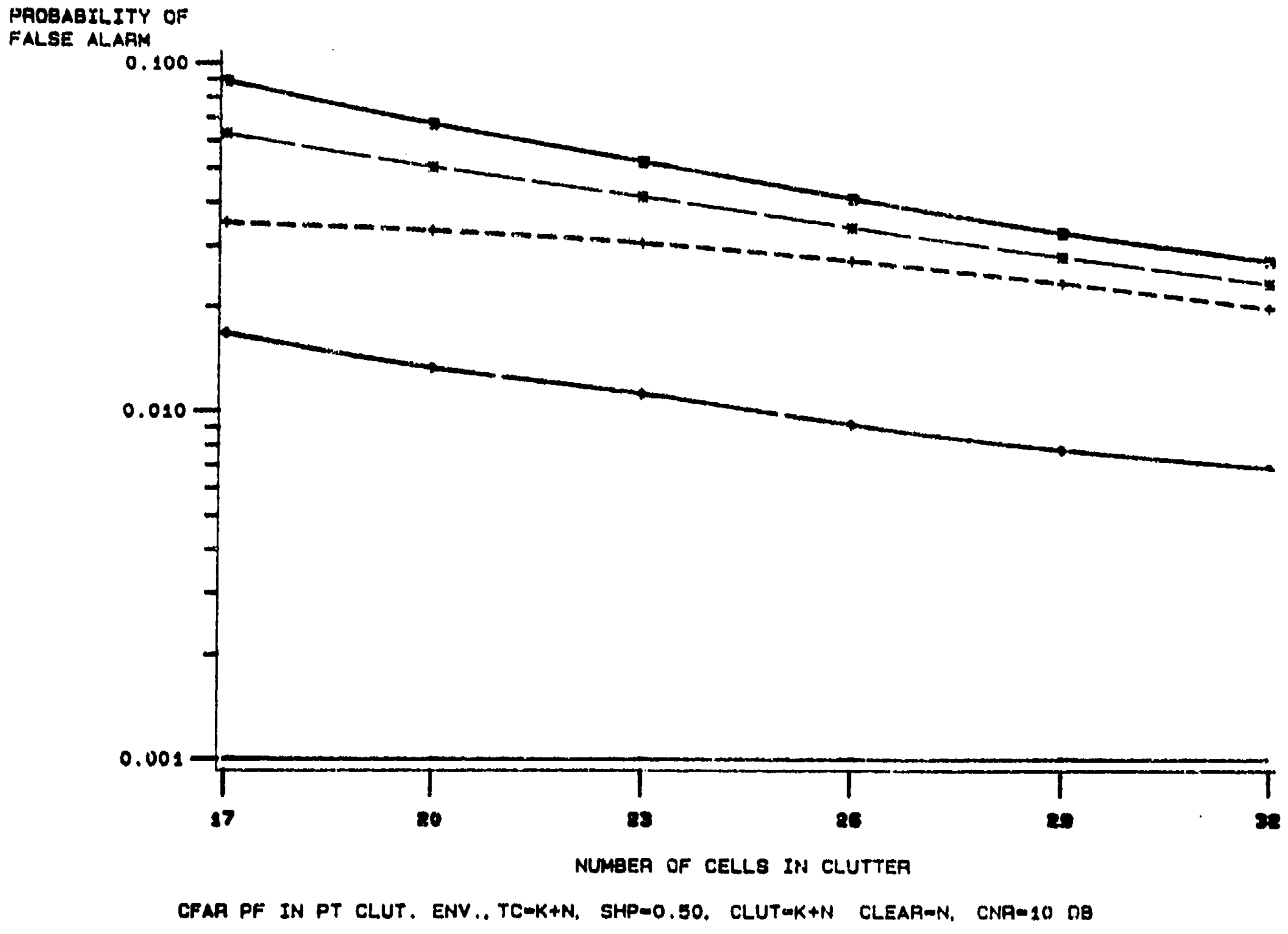


Figure 3-29: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 0.50. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

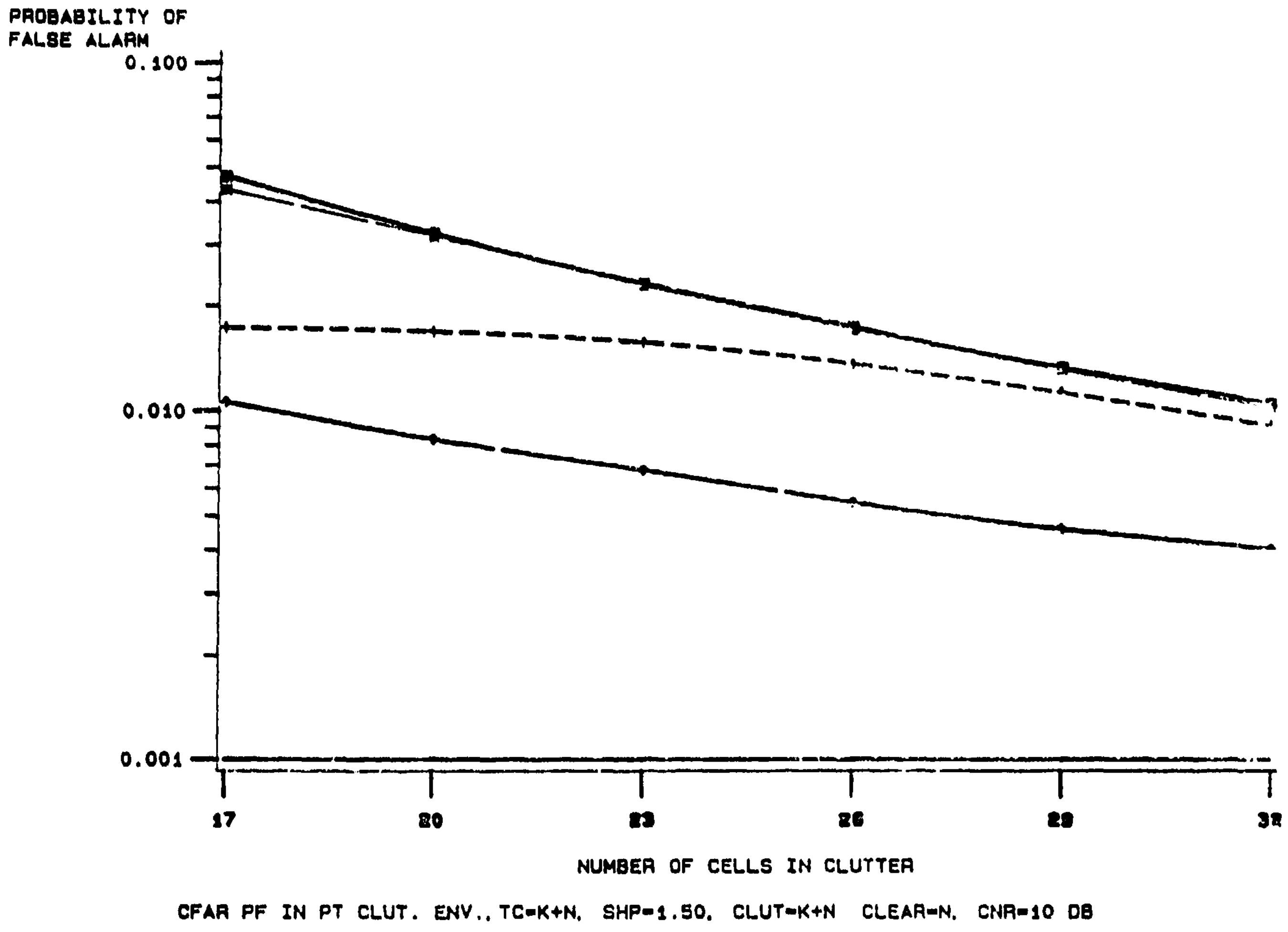


Figure 3-30: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 1.50. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

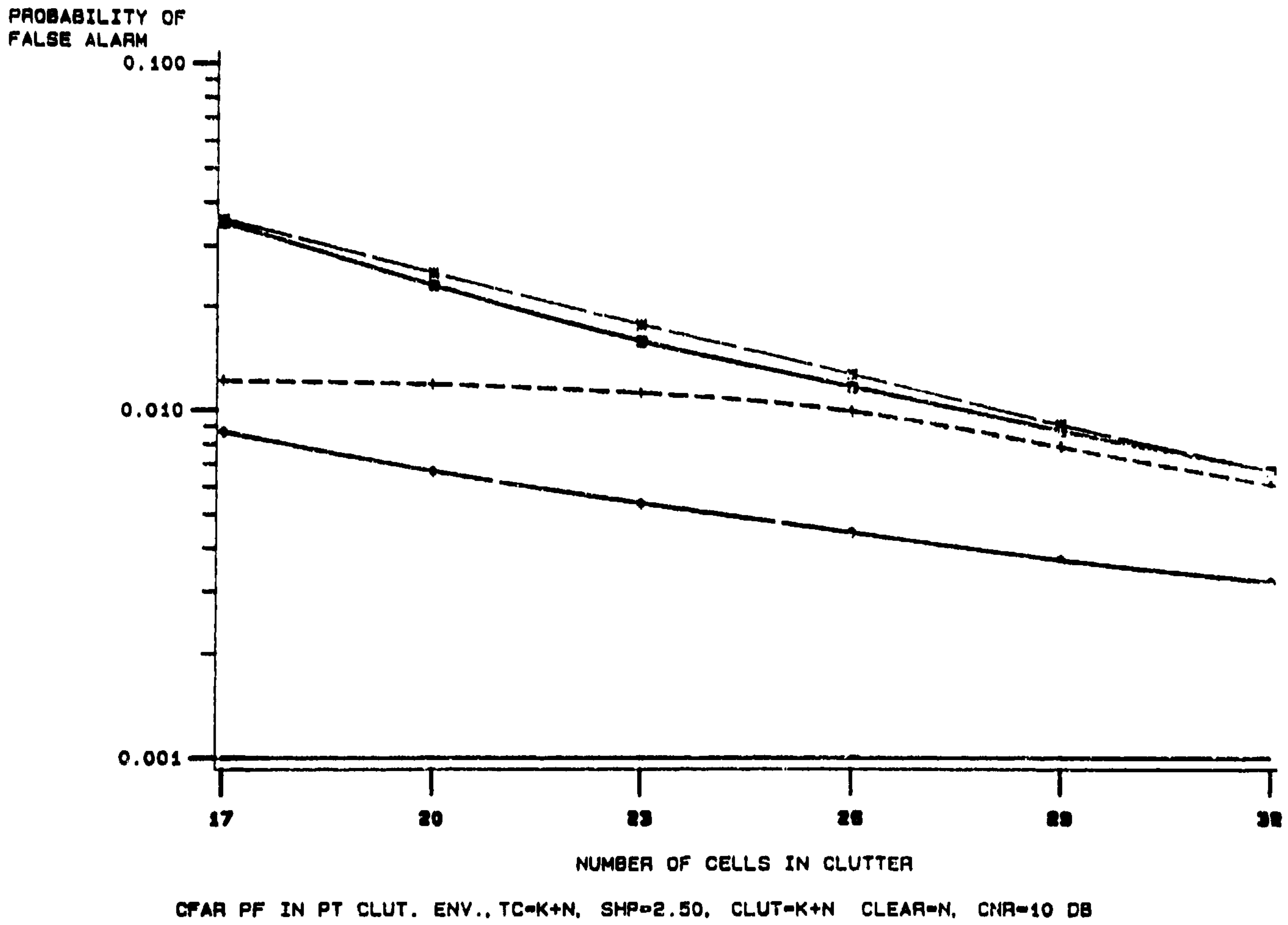


Figure 3-31: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 2.50. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

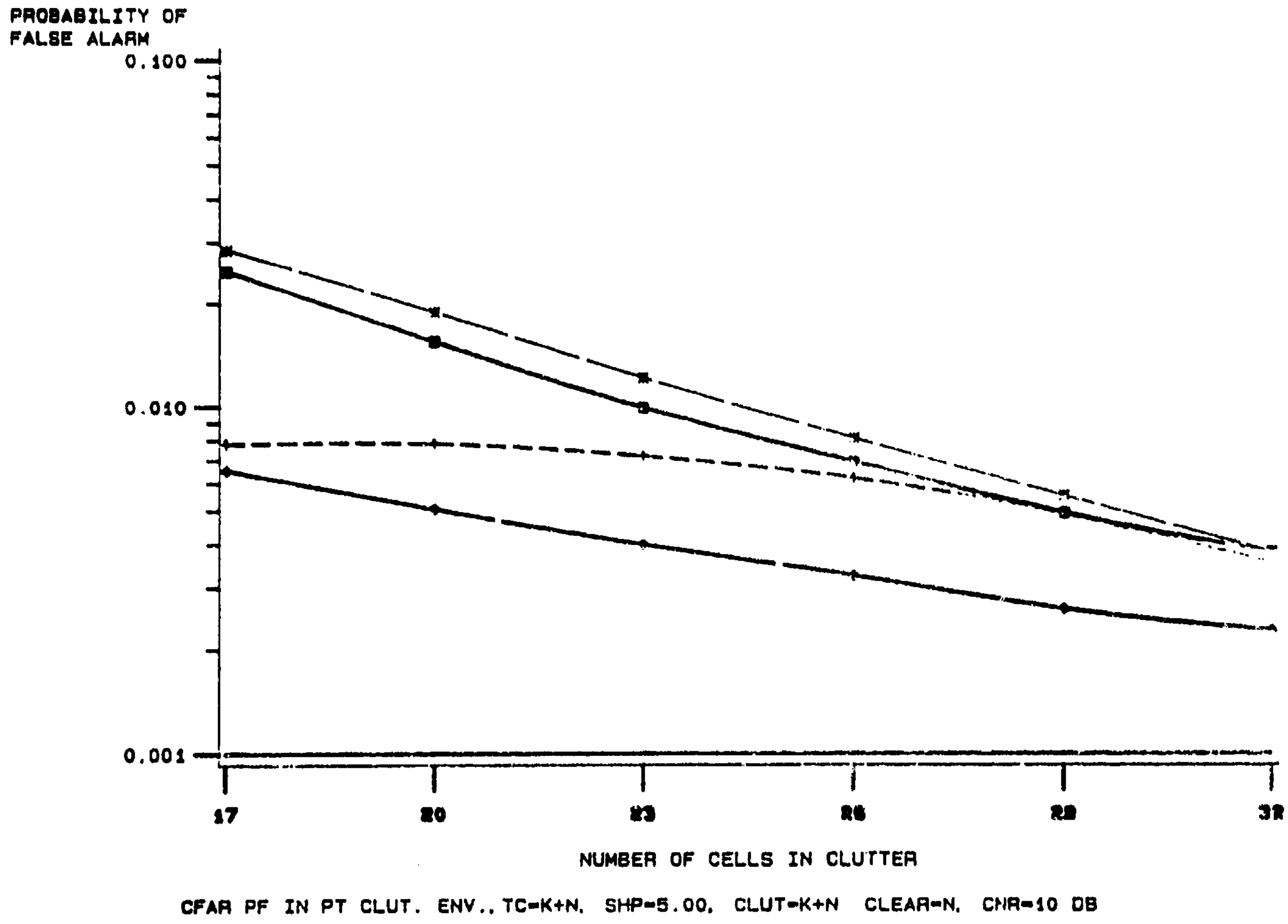


Figure 3-32: Probability of false alarm as a function of the number of cells in K-distributed clutter. Shape parameter is 5.00. CNR is 10 dB. Test cell is in clutter plus Gaussian noise.

Next, it was assumed that some of reference window cells were filled with K-distributed clutter plus Gaussian noise (K1+N) with CNR1=15 dB and the rest were also filled with K-distributed clutter plus Gaussian noise (K2+N) but with CNR2=5 dB. The test cell (TC) was assumed to be in K1+N. The P_f performances in this environment condition are shown in Figures 3-33 through 3-40 for all eight shape parameter.

3.2.2.3 Homogeneous Background with Different P_F Design Values

In an attempt to bring down the P_f level of the CFAR processors in K-distributed clutter-only environment down to the designed level ($P_f = 10^{-3}$), scale factors were varied for CFAR processors. These scale factors were calculated based on a Gaussian-only environment assumption, for P_F levels at 10^{-3}, 10^{-4},...,10^{-18}. With these scale factors, the P_F and P_D performance versus shape parameter, in K-distributed clutter-only homogeneous background condition are shown in Figures 3-41 through 3-46. SCR was equal to 10 dB. The results indicate that for small values of α , extremely low design P_f rates must be used to achieve the moderate P_f rates obtained in a Gaussian environment. Unfortunately, this causes P_d to degrade to unacceptably low levels. In other words, it is not practical to vary the threshold in a non-Gaussian environment to achieve the same false alarm rate achieved in a Gaussian environment, if the CFAR is designed using a Gaussian assumption.

These simulation results will be used to develop the decision rules for the ES CFAR system, to select the best CFAR processor with its parameters set at optimum values to maintain the desired constant false alarm probability while obtaining the highest possible probability of detection.

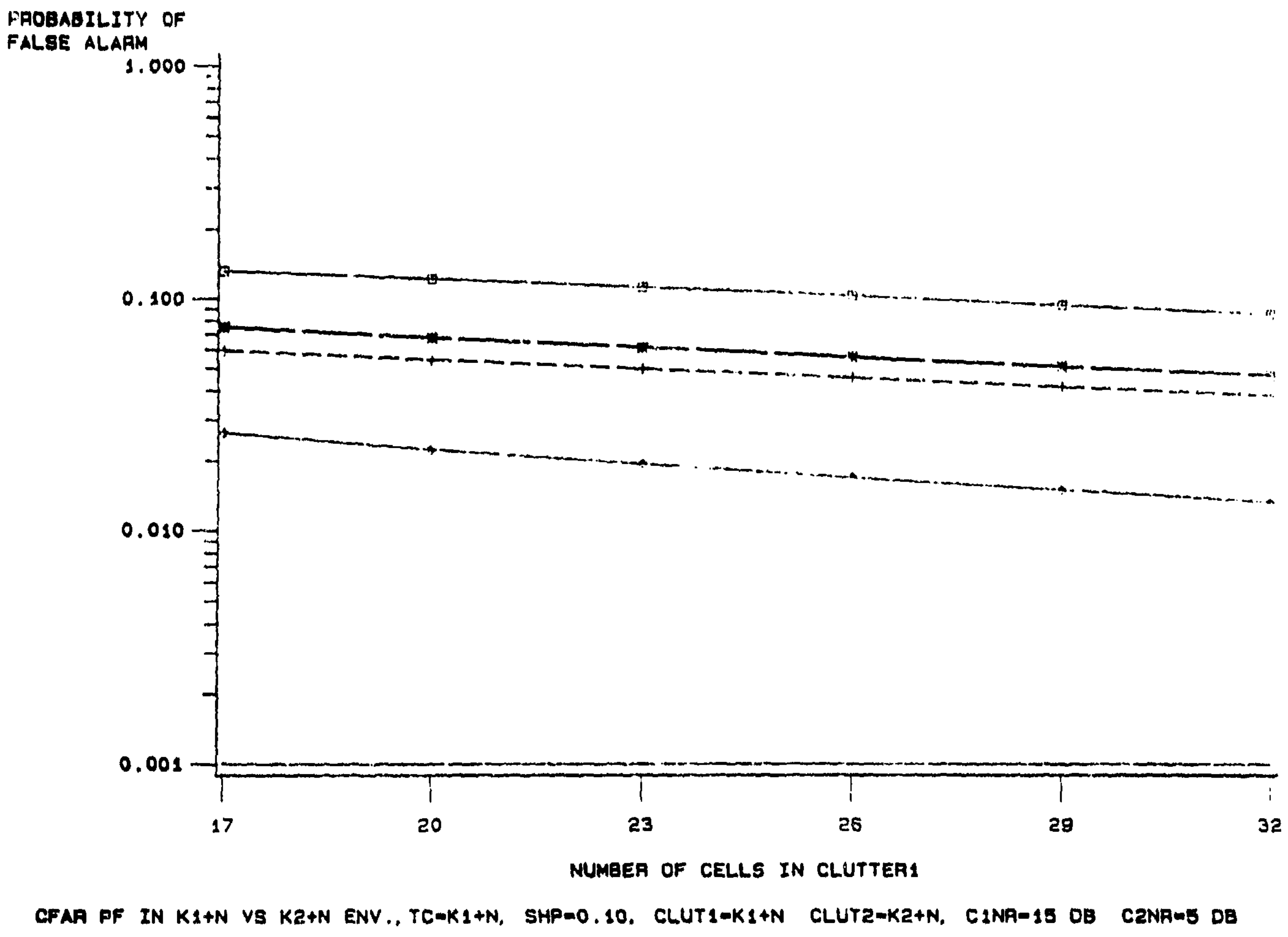


Figure 3-33: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 0.10. C1NR is 15 dB. C2NR is 5 dB

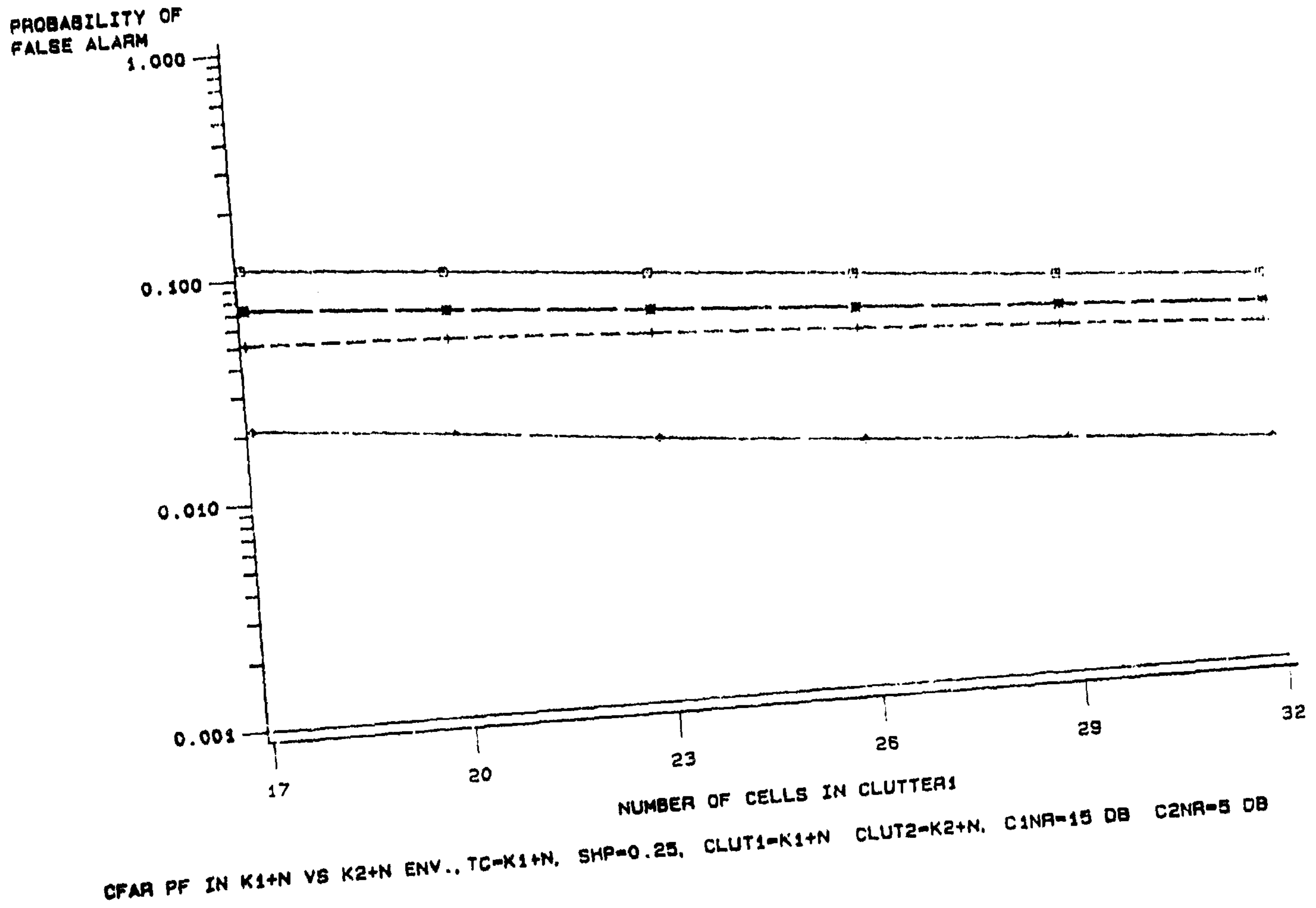


Figure 3-34: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 0.25. C1NR is 15 dB. C2NR is 5 dB.

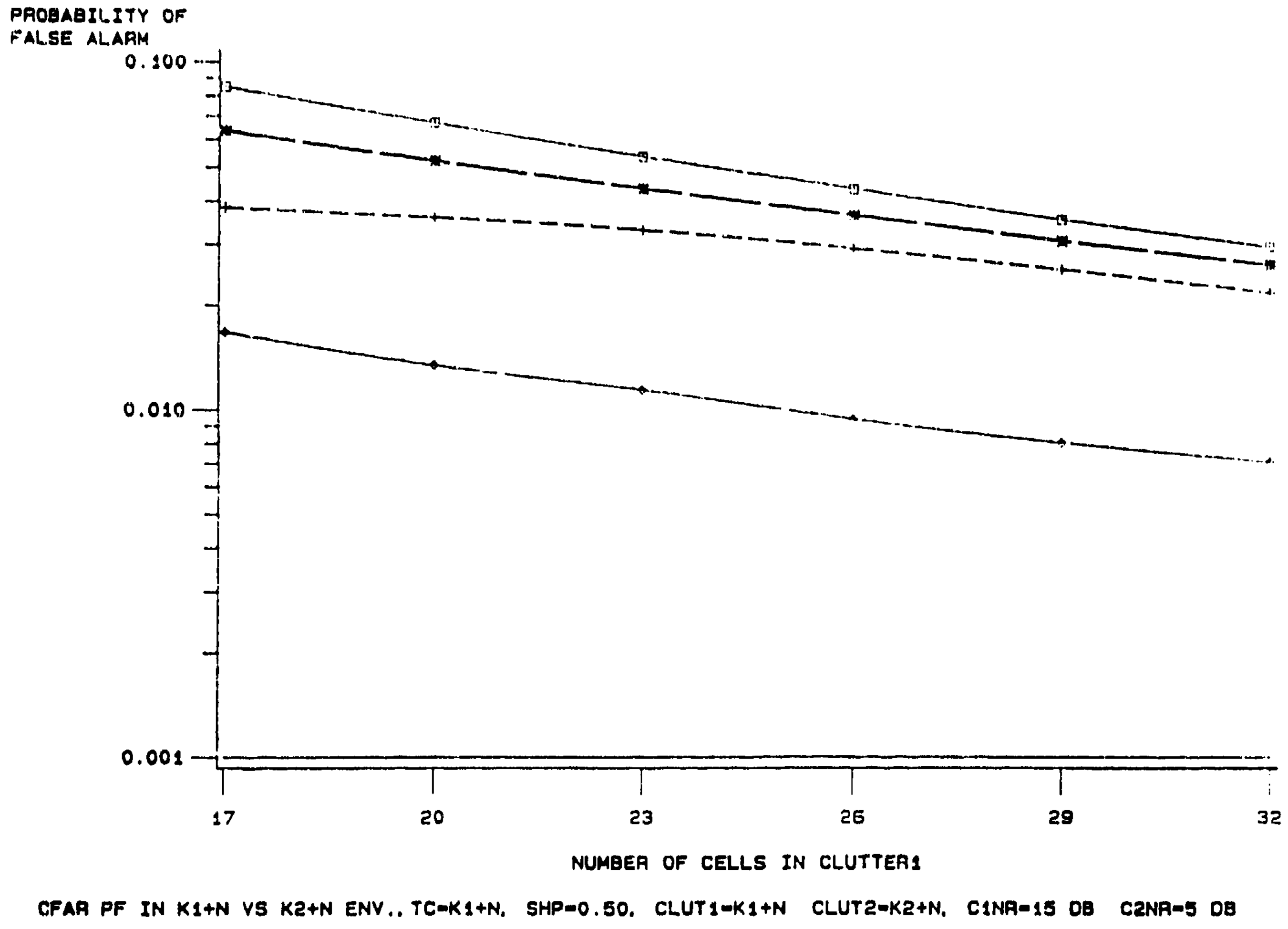


Figure 3-35: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 0.50. C1NR is 15 dB. C2NR is 5 dB.

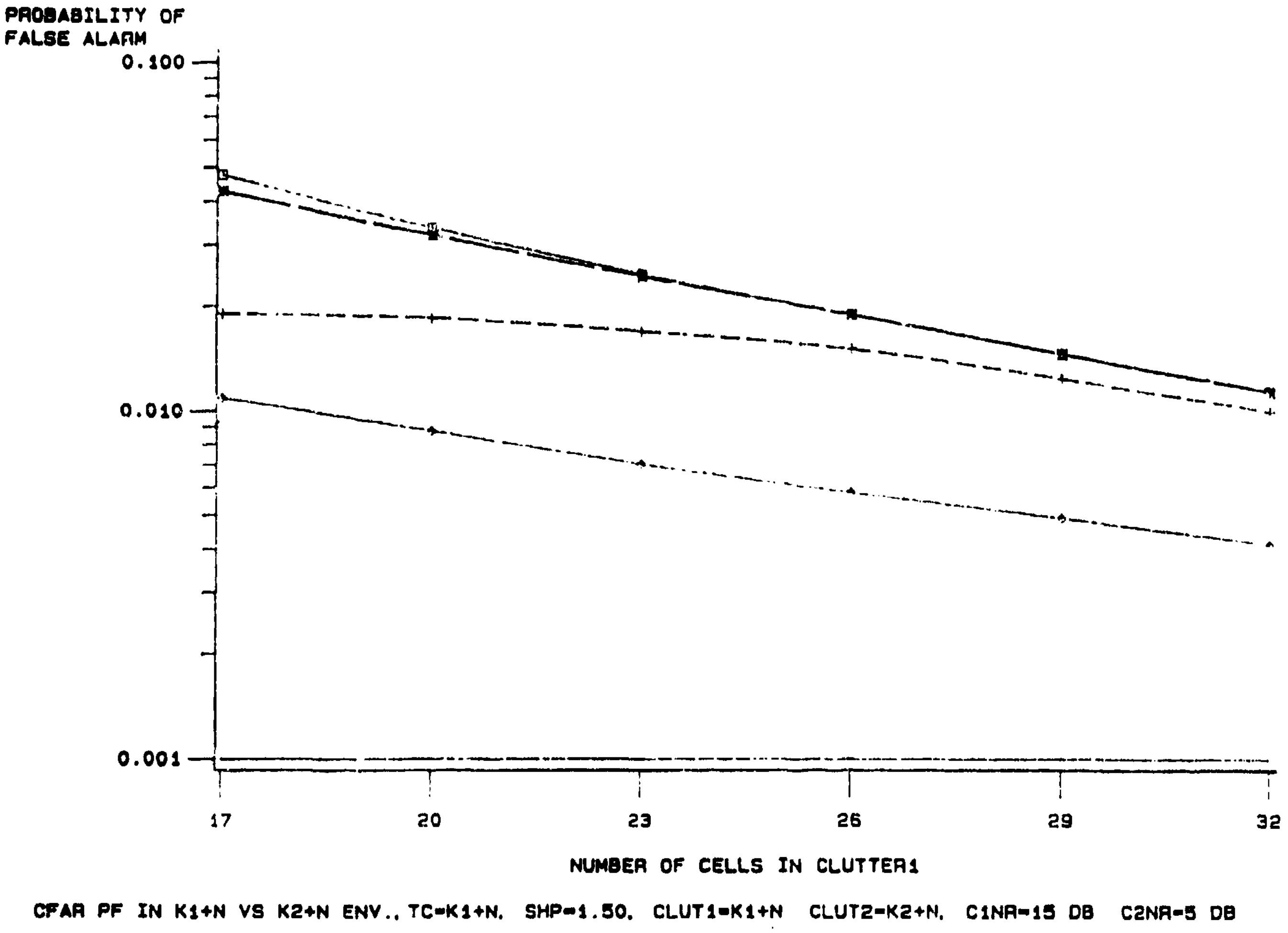


Figure 3-36: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 1.50. C1NR is 15 dB. C2NR is 5 dB.

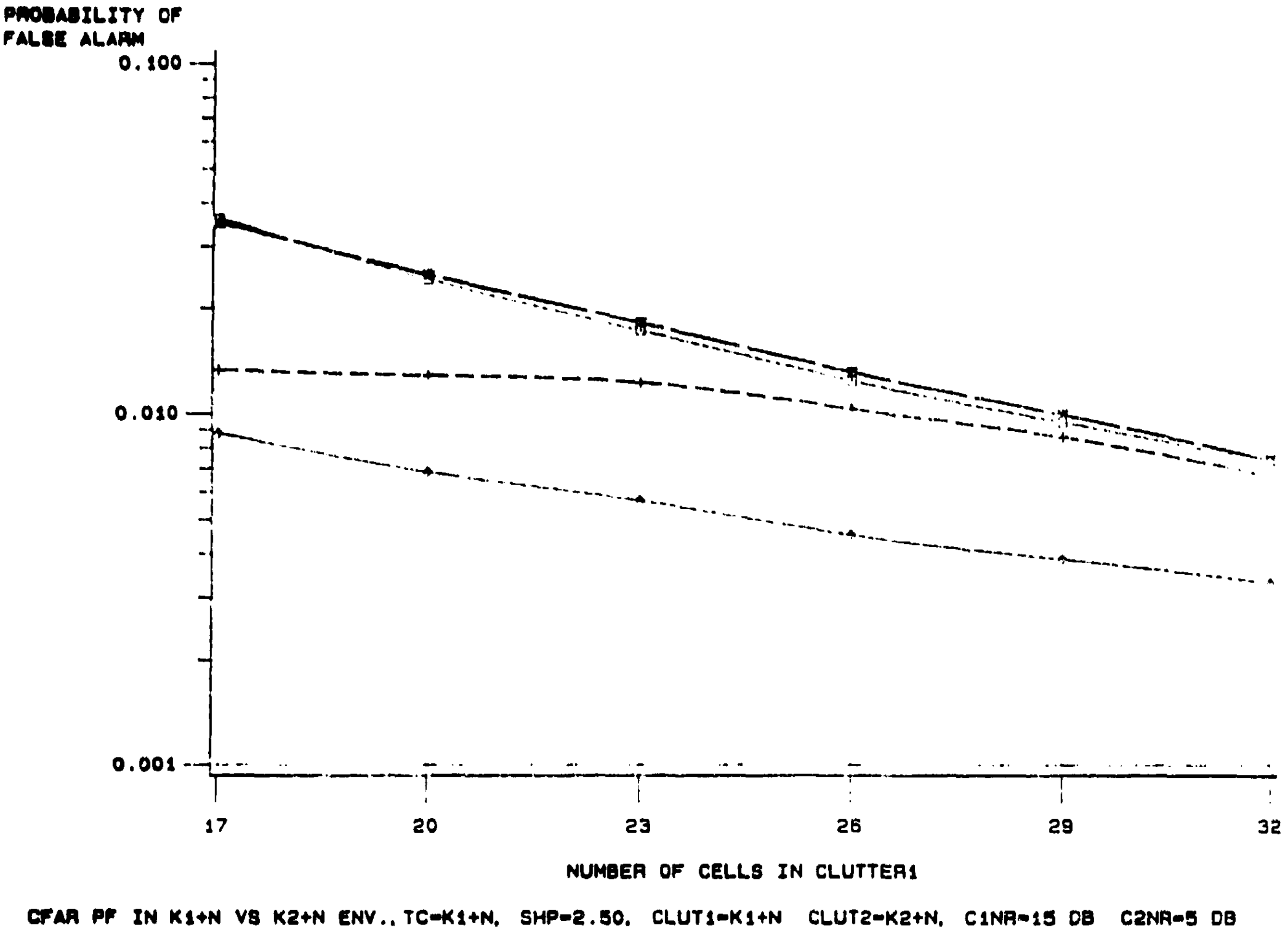


Figure 3-37: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 2.50. C1NR is 15 dB. C2NR is 5 dB.

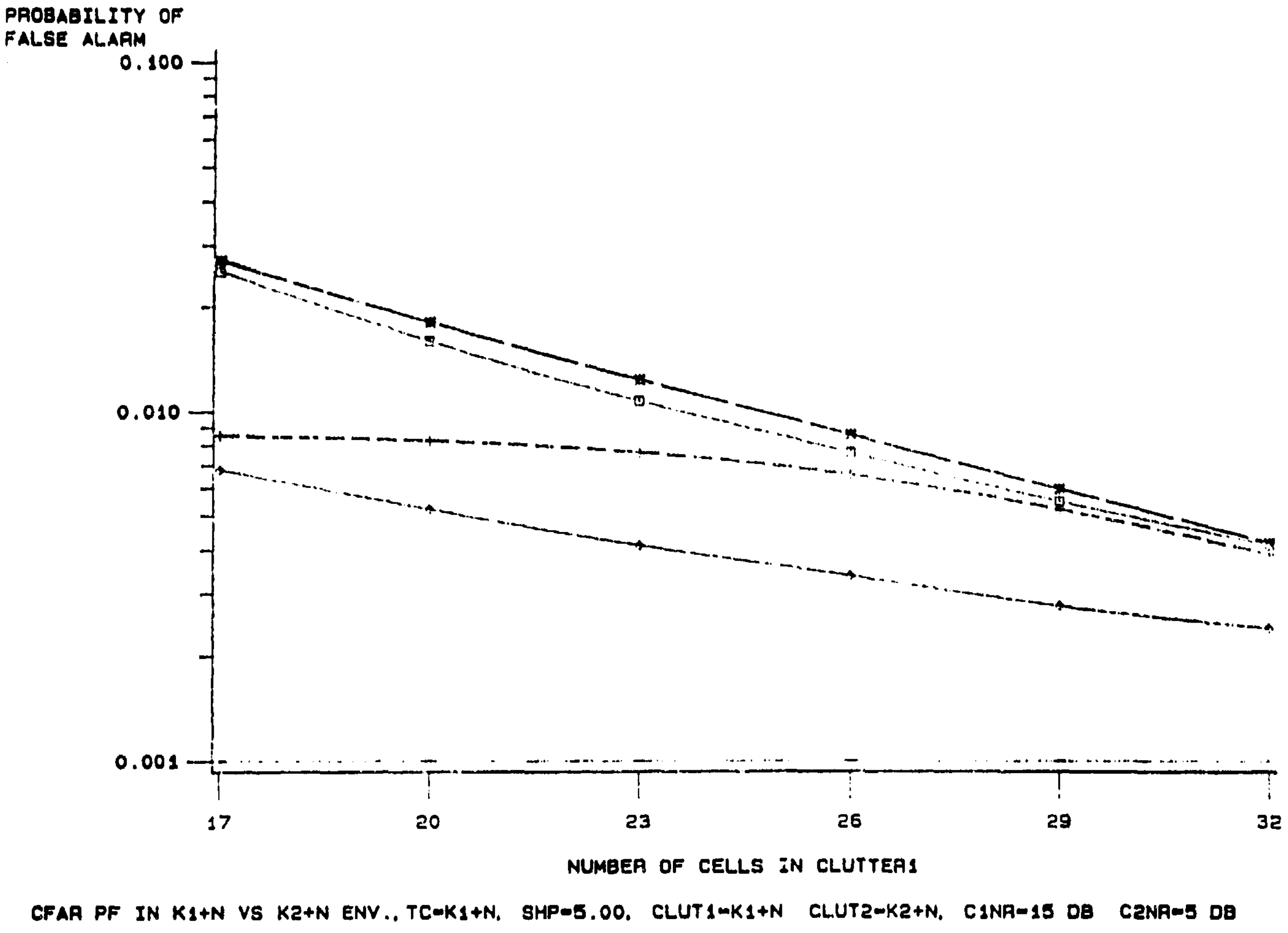


Figure 3-38: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 5.00. C1NR is 15 dB. C2NR is 5 dB.

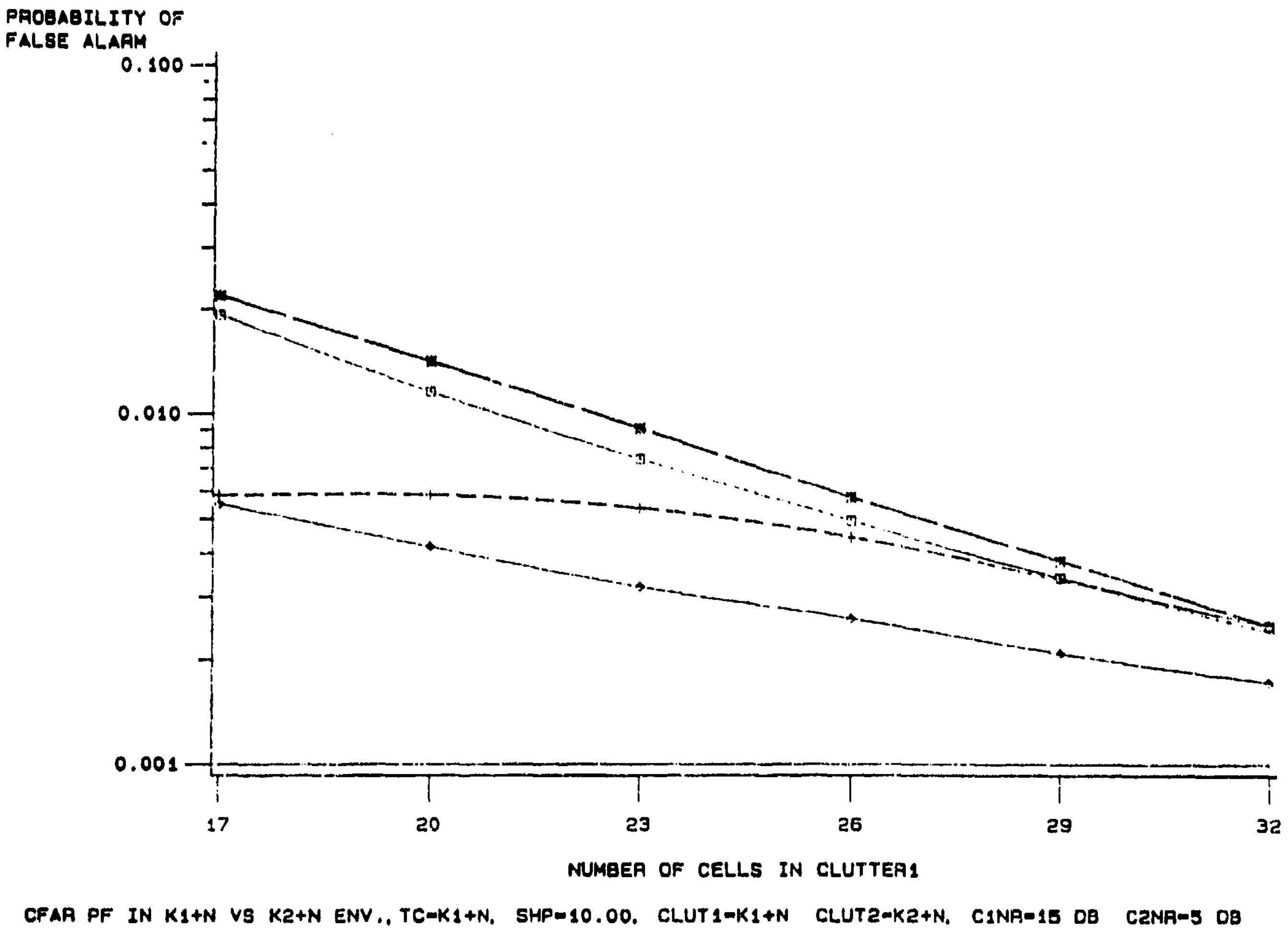


Figure 3-39: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 10.00. C1NR is 15 dB. C2NR is 5 dB.

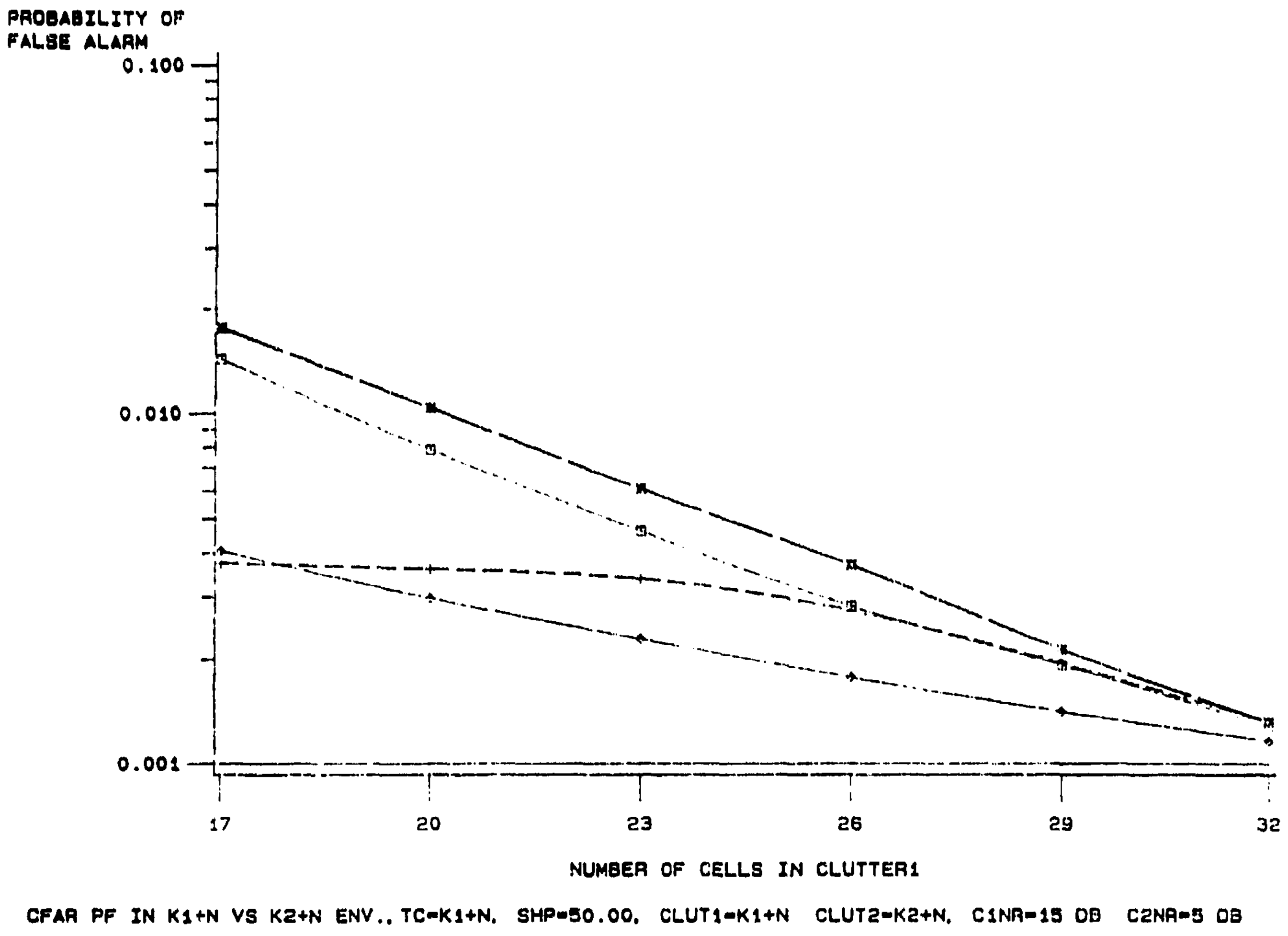


Figure 3-40: Probability of false alarm in an environment of two clutter regions plus thermal noise. Plotted as a function of the number of cells in clutter level 1. Test cell contains clutter level 1 plus noise. Shape parameter is 50.00. C1NR is 15 dB. C2NR is 5 dB.

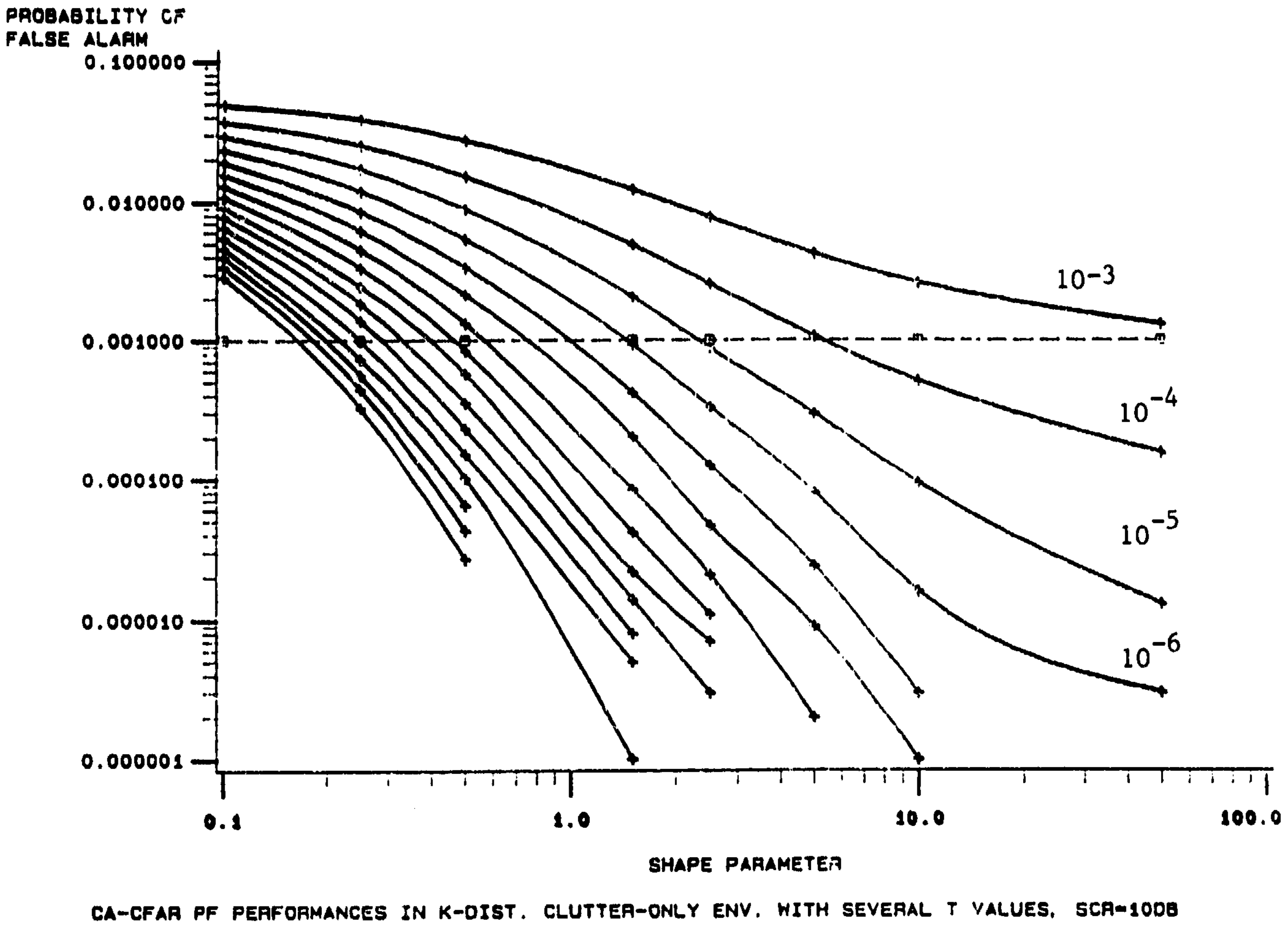


Figure 3-41: Actual probability of false alarm of CA CFAR in K-distributed clutter, parametric on the design value of P_f.

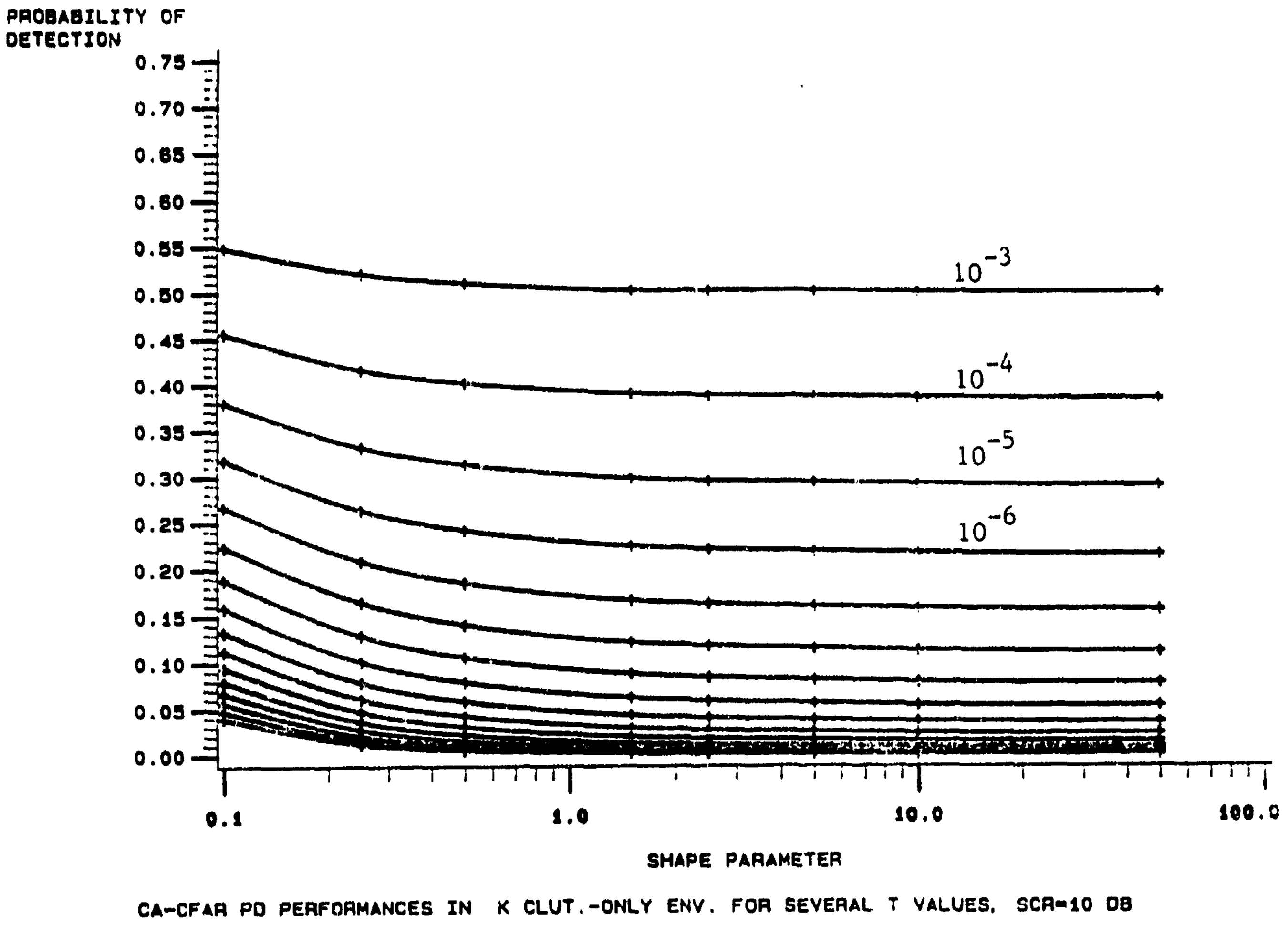


Figure 3-42: Probability of detection of CA CFAR in K-distributed clutter, parametric on the design value of P_f.

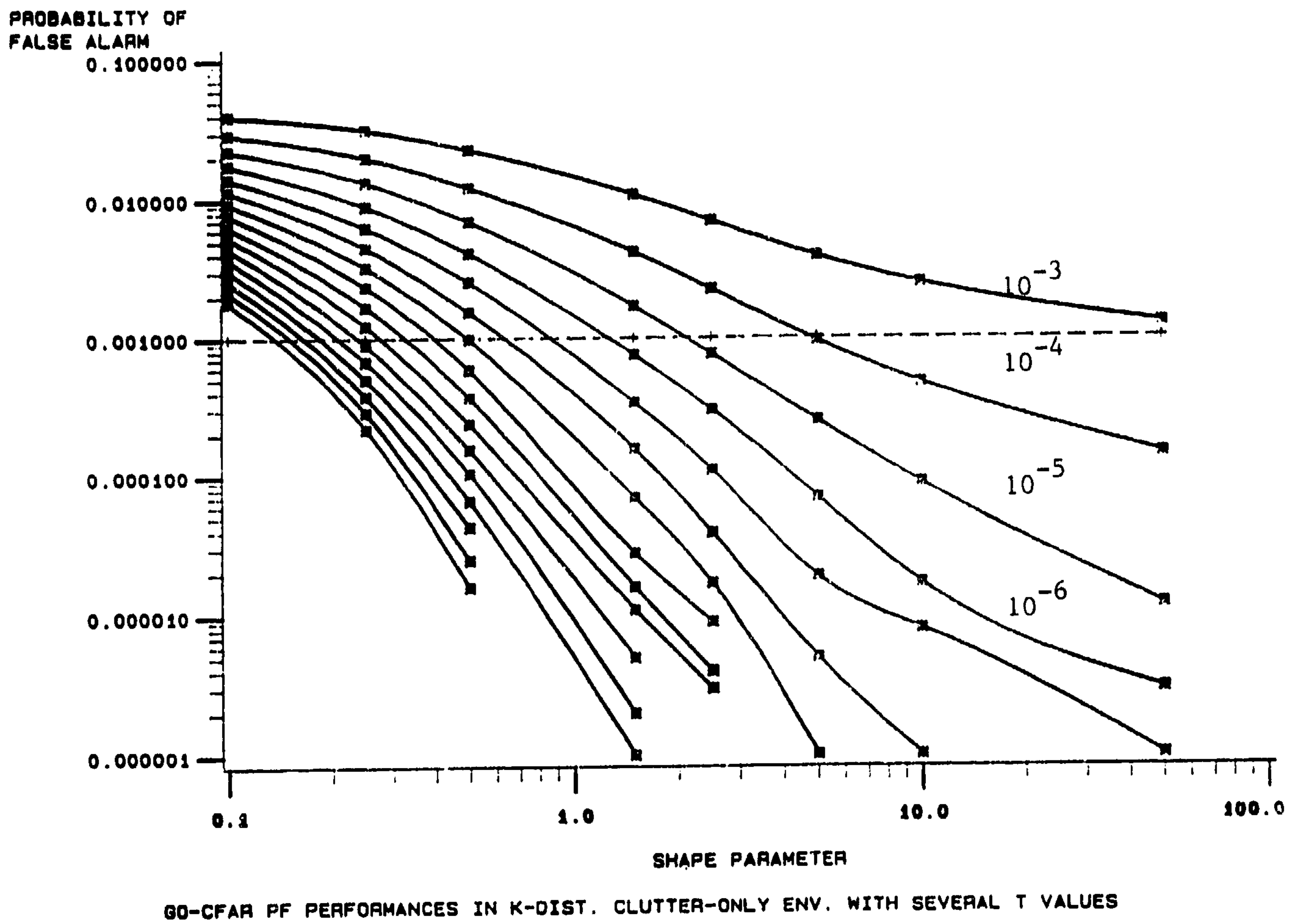


Figure 3-43: Actual probability of false alarm of GO CFAR in K-distributed clutter, parametric on the design value of P_f.

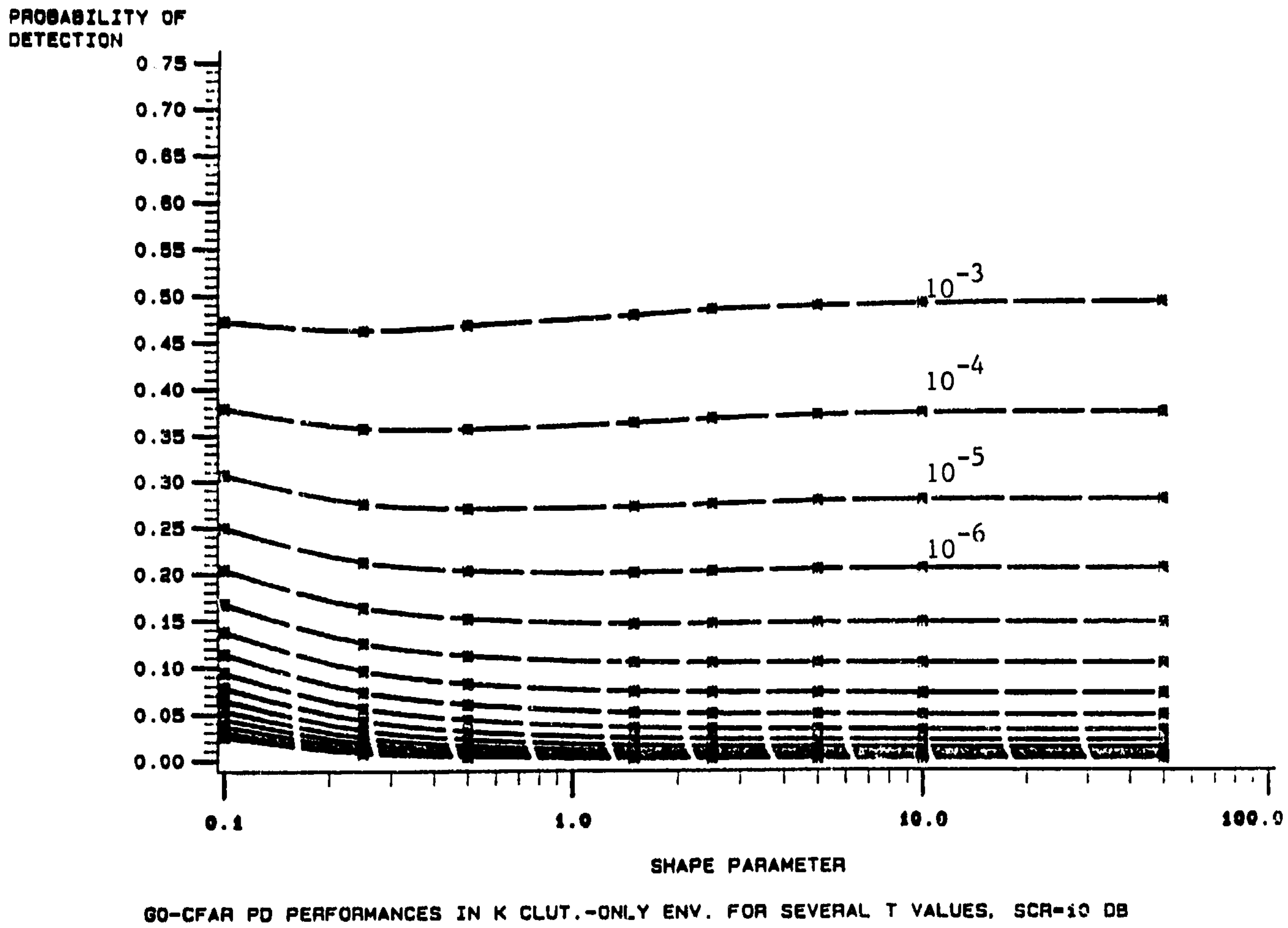


Figure 3-44: Probability of detection of GO CFAR in K-distributed clutter, parametric on the design value of P_f.

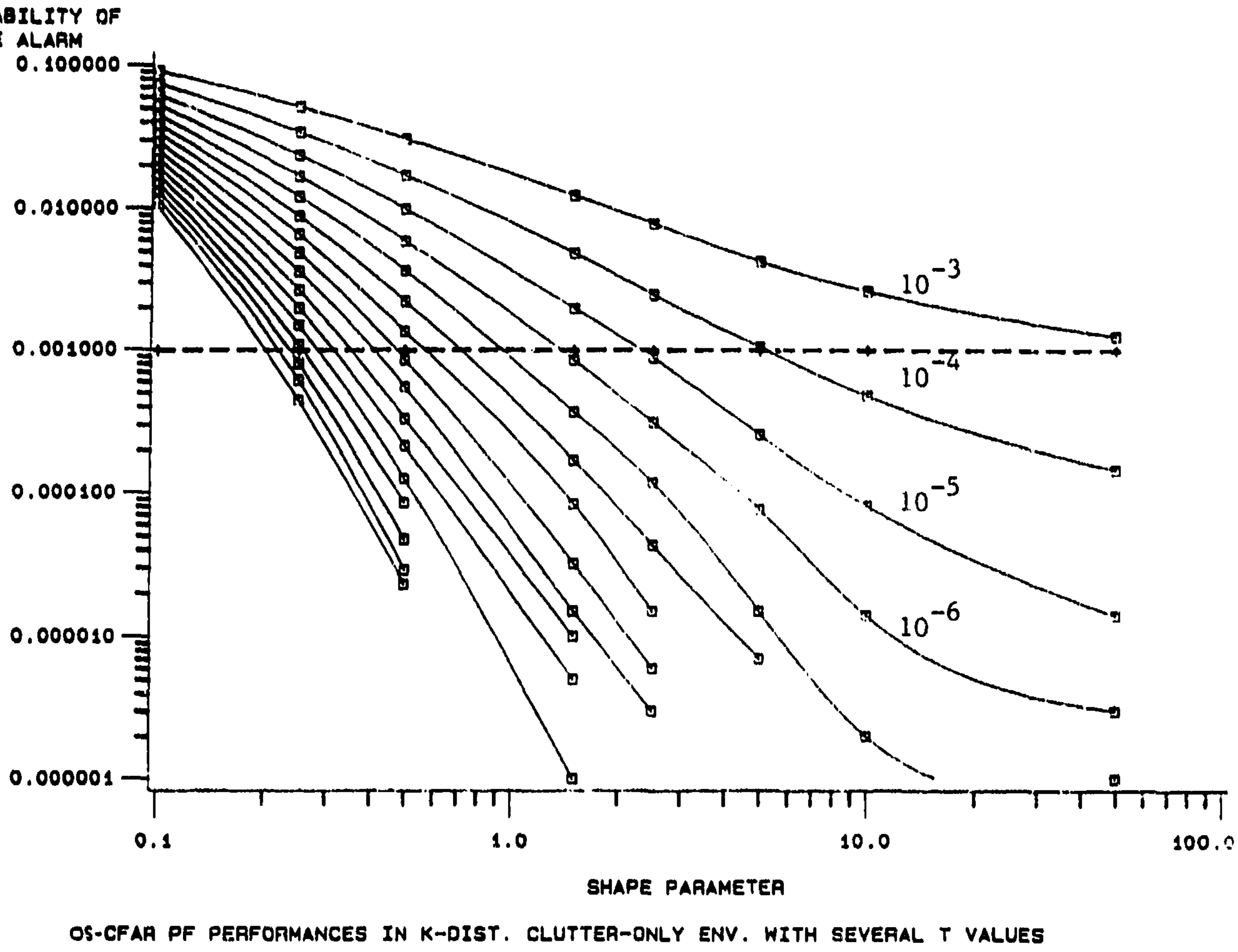


Figure 3-45: Actual probability of false alarm of OS CFAR in K-distributed clutter, parametric on the design value of P_f.

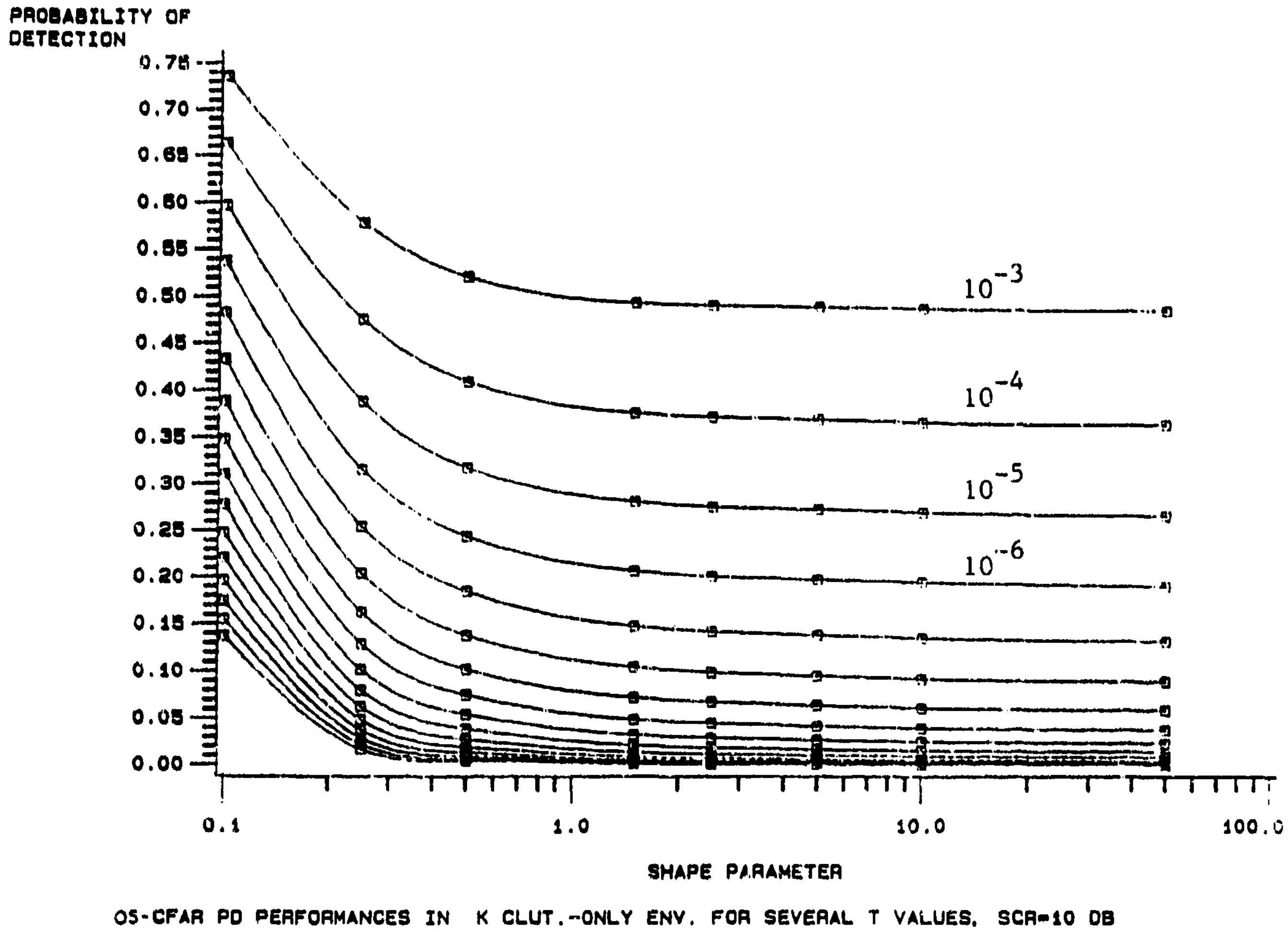


Figure 3-46: Probability of detection of OS CFAR in K-distributed clutter, parametric on the design value of P_f.

4.0 SUMMARY OF TECHNICAL PROGRAM STATUS

This section summarizes the current program status and identifies near term and longer range goals. In the long term this is a rather ambitious program which uses AI techniques in virtually all aspects of radar signal data processing. As these techniques become more widespread and the results become more accepted, many approaches will evolve into (more or less) fixed algorithms. Others may remain in the AI realm for quite some time since there will always be a need to operate in a dynamic environment which can never be fully anticipated or characterized. It is this ever changing environment, which normally requires human interaction, that gives impetus to AI techniques. Whenever the environment is known (deterministic) or can be completely characterized statistically, optimal strategies either exist or can be developed. As always, near optimal strategies may actually be of more value from an efficiency standpoint.

At the end of this current effort an expert system framework will be implemented with an example system which focuses on CFAR processing. This system will automatically examine the radar clutter environment, in ways similar to what a human would do, and then select the optimal or near-optimal CFAR which is judged most likely to be applicable. Notice that if the environment is classified correctly and the proper CFAR algorithms are available then the goal of maximizing the probability of detection while maintaining a constant false alarm rate will be achieved. It is unlikely that clutter will be classified correctly 100% of the time, partly because some environments are too similar and partly because environments exist that do not fit into any of the pre-determined classes. An ideal system would include all possible environments but this is obviously not practical. An appropriate CFAR algorithm needs to be included in the CFAR library for each anticipated environment. Currently, this library contains four algorithms: CA, GO, OS, and TM. These four, with variations in their respective parameters, can be utilized effectively in a large number of environments. At this time, additional CFARs will probably not be added under the existing contract. However, it is very likely that other CFARs will be added in the long term. The composition of the software under this contract is being decided based on a compromise which attempts to implement the most effective system given the available resources. The rest of this section discusses the various compromises envisioned for each software area.

Radar Data Sources. To evaluate various heuristics the system must be capable of creating the appropriate environment consisting of receiver noise, external noise, mixed clutter, and targets. Currently, receiver noise, Weibull clutter, log-normal clutter, discrete clutter and Swerling I targets can be generated. By the end of this contract K-distributed clutter will also be included. This data is currently generated in azimuth/range space. By the end of this contract Doppler space will also be added. In the long term, the need for other clutter and target types is anticipated, including dynamic high fidelity RCS target models which would enable the system to evaluate the use of adaptive waveforms for matched target illuminations. This should be a very effective technique for detecting low RCS targets in heavy mixed clutter.

In addition, by the end of this contract recorded data will be injected into the system. In the long term it is likely that more types of recorded data as well as real-time data will be needed. Additionally, the capability to utilize multi-dimensional CFAR's in range, azimuth, elevation and Doppler, would be desirable. The system which will be completed under this contract, will perform CFAR processing in the range domain only.

Environmental Inputs. Currently no environmental inputs have been implemented. Under this contract the user is expected to be capable of specifying regions of the simulated data to be area clutter such as land, trees, sea, etc. In the long term, information pertinent to weather should be incorporated. This would be useful in locating weather clutter, anomalous propagation, atmospheric layers, ducting environments, etc. Also the addition of topographical data, while complex, may become feasible at some point in the future.

CFAR Selection Rules. Only a few rudimentary rules currently exist. This area is the current focus of the effort and as such is not ready to be included in this report. By the end of this contract a comprehensive set of rules is expected to be implemented, based on the extensive literature search completed earlier in the program and the simulations that have been performed. In the long term, as more CFARs are added and after various experts have been exposed to this program, it is reasonable to assume that the CFAR selection rules data base will grow.

CFAR Pre-processor. Currently the CFAR pre-processor segments each radial into regions made up of contiguous range samples which are similar statistically. This allows different CFARs to operate on different parts of each radial of data. Several edge detection algorithms have been identified, including some used in digital image processing, and these are currently being tested. Spot detection algorithms from digital image processing are also being considered to identify and tag isolated discontinuities which arise from targets, discrete clutter, noise glitches, etc. These points will be ignored by the CFARs except when they are actually in a test cell. Additionally, the pre-processor currently computes a list of statistics for each region including mean, variance, skewness, kurtosis, mode, and correlation coefficient. These statistical estimates are used by the CFAR selection rules to help select the appropriate CFAR. In the long term the pre-processor can be expanded to implement other non-linear transformations to facilitate optimal detection for specific environments.

CFAR Library. The status of the CFAR library was summarized at the beginning of this section. In addition to those comments, any of the CFARs can be selected as the baseline CFAR. All CFARs are currently designed assuming an exponential (power) clutter distribution. Under this contract the necessary code will be implemented to adjust the desired false alarm rate which is input to the ES CFARs to simulate other tail shapes. This empirically-derived adjustment will be based on the tail estimator which is part of the CFAR pre-processor. In the long term, as additional optimal receivers and clutter classification schemes are developed, the CFAR library is expected to expand to include CFARs which assume a variety of tail shapes (ie: not just the Gaussian assumption).

CFAR Post-processor. This is the software area where data fusion occurs and where the resultant candidate detections are processed using AI techniques to further control false alarms. The pre-processor attempts to standardize the radar data into manageable environments. The CFAR selection rules then attempt to select the most applicable CFAR. When the environment adequately matches the CFAR design assumptions, a constant false alarm rate will be maintained. In general, perfect knowledge is not available and perfect decisions are not always made (experts make mistakes), so as a result the false alarm rate will vary. By monitoring the actual false alarm rate and by utilizing confidence levels estimated in the pre-processor, the post-processor will trim the candidate detections when necessary to better maintain a constant false alarm rate. Currently the post-processor is essentially a stub. First-order false alarm reconciliation is expected to be included by the end of the contract. In the long term, the post-processor would include such functions as track feedback, target library correlation, false alarm library correlation, and further tests to separate targets from false alarms.

Performance Measure. Currently, procedures are implemented to estimate P_d and P_f for both the baseline and ES CFAR paths. For selected environments and recorded data the desired P_f for the baseline CFAR can be adjusted until the actual P_f for the baseline and ES CFARs are equal. Target RCS can then be adjusted until the P_d is equal for both. The dB change in target RCS will provide the desired quantified performance measure. Of course the P_f for both channels prior to any adjustment will also serve as a performance measure, but it is not considered to be a stand-alone measure since it does not take into account the P_d. This process is expected to be automated under this contract. In the long term it is expected that other performance measures would be put forth as more people become exposed to this program.

Displays. Displays which are currently implemented include PPIs for both the baseline CFAR channel and the ES CFAR channel, as well as a comparison PPI. Current displays also include the current radial of radar data vs. range, the estimated P_d and P_f for both channels vs. time, and the simulated clutter map. While its graphical user interface is elaborate, G2 has limited display capability for plotting data. PV-WAVE 44 is being used to produce the power versus range plots as shown in Section 2. Newer versions of G2 and PV-WAVE will be evaluated in coming weeks to automate the various displays. In the long term, additional displays that illustrate variations with altitude should be included as well as target track displays.

REFERENCES

[1] P.P. Gandhi and S. A. Kassam, "Analysis of CFAR processors in non homogeneous background," IEEE Transactions on Aerospace and Electronic Systems, vol. 24, pp. 427-445, July 1988.

[2] Press, W.H., Flannery, B.P. Teukolsky, S.A., and Vetterling, W.T., 1988: "Numerical Recipes in C," Cambridge University Press, New York, 735 pp.

[3] Afifi, A.A. and S.P. Azen 1972: "Statistical Analysis, A Computer Oriented Approach," Academic Press, New York, 366 pp.

[4] H. M Finn and R. S. Johnson, "Adaptive detection with threshold control as a function of spatially sampled clutter-level estimates," RCA Review, vol. 29, pp. 414-464, September 1968.

[5] R. Nitzberg, "Analysis of the arithmetic mean CFAR normalizer for fluctuating targets," IEEE Transactions on Aerospace and Electronic Systems, vol. 14, pp. 44-47, January 1978.

[6] R. Nitzberg, Application of invariant hypothesis testing techniques to signal processing. PhD thesis, Syracuse University, 1970.

[7] L. L. Scharf and D. W. Lyte, "Signal detection in Gaussian noise of unknown level: An invariance application," IEEE Transactions on Information Theory, vol. 17, pp. 404-411, July 1971.

[8] V. G. Hansen, "Constant false alarm rate processing in search radars," in International Radar Conference, (London), pp. 325-332, IEE, 1973.

[9] J. D. Moore and N. B. Lawrence, "Comparison of two CFAR methods used with square law detection of Swerling I targets," in International Radar Conference, pp. 403-409, IEEE, 1980.

[10] J. Trickard and G. M. Dillard, "Adaptive detection algorithm for multiple target situations," IEEE Transactions on Aerospace and Electronic Systems, vol. 13, pp. 338-343, July 1977.

[11] M. Weiss, "Analysis of some modified cell-average CFAR processors in multiple target situations," IEEE Transactions on Aerospace and Electronic Systems, vol. 18, pp. 102-113, January 1982.

[12] V. G. Hansen and J. H. Sawyers, "Detectability loss due to Greatest of selection in a cell averaging CFAR," IEEE Transactions on Aerospace and Electronic Systems, Vol. 16, pp. 115-188, January 1980.

[13] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," IEEE Transactions on Aerospace and Electronic Systems, vol. 19, pp. 608-621, July 1983.

[14] G. V. Trunk, "Range resolution of targets using automatic detectors," IEEE Transactions on Aerospace and Electronic Systems, vol. 14, pp. 750-755, September 1978.

[15] E. Jakeman and P. N. Pusey, "A model for non-Rayleigh sea echo," IEEE Transactions on Antennas and Propagation, pp. 806-814, 1976.

[16] S. Watts, "Radar detection prediction in sea clutter using the compound K-distribution model," IEE Proceedings, Part F, vol. 132, pp. 613-620, December 1985.

[17] S. Watts, "Radar detection prediction in K-distributed sea clutter and thermal noise," IEEE Transactions on Aerospace and Electronic Systems, vol. 23, pp. 40-45, January 1987.

[18] B. C. Armstrong and H. D. Griffiths, "CFAR detection of fluctuating targets in spatially correlated K-distributed clutter," IEEE Proceedings, Part F, vol. 138, pp. 139-152, April 1991.

[19] R. J. Larsen and M. L. Marx, An introduction to mathematical statistics and its applications. New York: Prentice-Hall. 1986.

[20] Z. A. Karian and E. J. Dudewicz, Modern statistical, systems, and GPSS simulation. New York: Computer Science Press, 1991.

Appendix A

Summary of CFAR Literature Search

CFAR ALGORITHMS

The literature survey, discussed in the previous section, provided numerous references for many CFAR articles. A list of the abbreviations for the CFAR algorithms encountered in the literature search is given in Table 1. For convenience in associating an author or citation with a CFAR algorithm, a cross reference table is presented in Table 2. In the following subsections, a synopsis of CFAR algorithms is presented and a candidate set of CFAR algorithms for selection by AI logic is derived.

SUMMARY OF IMPORTANT CFAR ALGORITHMS

The typical CFAR processing algorithm is shown in Figure 1. For pulsed Doppler applications, the reference cells in a sliding window are used to estimate a background noise/clutter level. The power in the test cell is compared to a threshold, determined from the background clutter level and the desired false alarm rate. If the power in the test cell exceeds the threshold, a detection is declared. The sliding window can incorporate data from adjacent ranges, Doppler cells, or adjacent azimuths. In some CFARs, the cells adjacent to the test cell (guard cells) are not used in the CFAR processing to avoid masking effects that could occur in multiple target situations or when the target extends through several range gates. For convenience, few references are given in the following subsections. The cross reference Table 2 can be used to determine literature citations for each type of CFAR.

Binary Integration

Binary integration (also called binary detection) is a simple detection method with relatively good performance. Binary integration has the advantage of good performance in non-Gaussian clutter environments. A set of N radar pulses are used in a double threshold process. In the first threshold, the amplitude is compared to a threshold. In the second threshold, the number of times the first threshold was exceeded is compared to a value M, less than N. If the sum of first threshold crossings exceeds M, a target is declared. The clutter envelopes D_i, $i = 1,2,...,N$ are assumed to be independent and identically distributed. An optimal value of M can be taken to be about $1.5N^{0.5}$ for a wide variety of conditions. Given M and the desired P_f, the threshold T can be determined if the pdf of the D_i is known. The pdf is assumed to be exactly known. A Rayleigh pdf is often

Table 1. List of CFAR algorithm abbreviations.

Abbreviation	Alias	CFAR Algorithm
BI		Binary Integration
CA		Cell Averaging
CAGO	GO	Cell Averaging Greatest Of
CCA		Censored (excision) Cell Averaging
CGO		Censored Greatest Of
CM		Clutter Map
CMLD		Censored Mean Level Detector
CVI		Censored Video Integrator
DDL-GLR		Doppler Domain Localized Generalized Likelihood Ratio
DF	NP	Distribution Free
GCA		Generalized Cell Averaging
GLR		Generalized Likelihood Ratio
GO		Greatest Of
GST		Generalized Sign Test
GTL-CMLD		Generalized Two-Level Censored Mean Level Detector
HCE		Heterogeneous Clutter Estimating
LOG		Logarithmic
Log-t		Log-t detector
MGO		Modified Greatest Of
MGST		Modified Generalized Sign Test
MLD		Mean Level Detector
MW		Mann-Whitney
MX-CMLD		Censored Mean Level Detector
MX-LCD		Maximum Linear Combination Detector
MX-MLD	GO	Maximum Mean Level Detector
MX-OSD		Ordered Statistic Detector
NP	DF	Nonparametric
OS	OSD	Ordered Statistic
OSD	OS	Ordered Statistic Detector
P		Polarimetric
SLC		Side Lobe Cancelling
SMI		Sample Matrix Inversion
SO		Smallest Of
SSA		Scan by Scan Averaging
TM		Trimmed Mean
WH		Weber-Haykin

Table 2. Cross reference table between CFAR algorithms and literature references.

CFAR	Author	Year
BI	Trunk	1983
BI	Dillard	1974
CA	Shore, et al.	1991
CA	Himonas, et al	1990
CA	Sekine, et al.	1989
CA	Gandhi, et al.	1988
CA	Conte, et al.	1988
CA	Bucciarelli	1987
CA	Trunk	1983
CA	Weiss, et al.	1982
CA	Trunk	1978
CA	Trunk, et al.	1970
CA	Finn, et al.	1968
CA -mult. bgd. est.	Barkat, et al.	1988
CA -mult. sensor	Barkat, et al.	1991
CCA	Shor, et al.	1991
CCA	Conte, et al.	1989
CCA	Trunk	1978
CCA	Rickard, et al.	1974
CGO	Al-Hussaini	1988b
CM	Nitzberg	1986
CM	Trunk	1983
CMLD	Ritcey	1986
CVI	Levanon	1990
DDL-GLR	Wang, et al.	1991
DDL-GLR	Wang, et al.	1990
DF	Trunk	1983
DF	Dillard, et al.	1970
Excision	Goldman	1990
Excision	Goldman, et al.	1988
GCA	Himonas, et al.	1990
GCMLD	Himonas, et al.	1989
GLR	Wang, et al.	1991
GLR	Cai, et al.	1991
GO	Ritcey	1990
GO	Elias-Fuste, et al.	1990
GO	Al-Hussaini	1988b
GO	Al-Hussaini	1988a
GO	Gandhi, et al.	1988
GO	Bucciarelli	1987
GO	Weiss, et al.	1982
GO	Hansen, et al.	1980
GO -mult. bgd. est.	Barkat, et al.	1988
GLR	Kelly	1986

Table 2. Cross reference table between CFAR algorithms and literature references. (Continued)

CFAR	Author	Year
GST	Trunk, et al.	1974
GST	Hansen, et al.	1971
GTL-CMLD	Himonas, et al.	1990
HCE	Finn	1986
LOG	Novak	1980
LOG	Hansen, et al.	1972
Log-t	Goldstein	1972
MGO	Bucciarelli	1987a
MGO	Bucciarelli	1987b
MGST	Trunk, et al.	1974
MLD	Ritcey, et al.	1991
MLD	Rickard, et al.	1977
MLD	Dillard	1974
MLD	Steenson	1968
MSMI	Cai, et al.	1991
MW	Hansen, et al.	1971
MX-CMLD	Ritcey, et al	1991
MX-LCD	Ritcey, et al.	1991
MX-OSD	Ritcey, et al.	1991
OS	Shor, et al.	1991
OS	Elias-Fuste, et al.	1990
OS	Levanon, et al.	1990
OS	Saniie, et al.	1990
OS	Levanon, et al.	1990
OS	Lei, et al.	1989
OS	Al-Hussaini	1988c
OS	Blake	1988
OS	Rohling	1983
Polarimetric	Wanielik, et al.	1990b
Polarimetric	Wanielik, et al.	1990a
Polynomial	Nitzberg	1973
Range-azimuth	Martin	1976
SMI	Cai, et al.	1991
SMI	Wang, et al.	1990
SO	Al-Hussaini	1988a
SO	Gandhi, et al.	1988
SO	Weiss, et al.	1982
SO -mult. bgd. est.	Barkat, et al.	1988
SSA	Lops, et al.	1989
Sign test	Hansen, et al.	1971
TM	Gandhi, et al.	1988
WH	Levanon, et al.	1990
WH	Weber, et al	1985

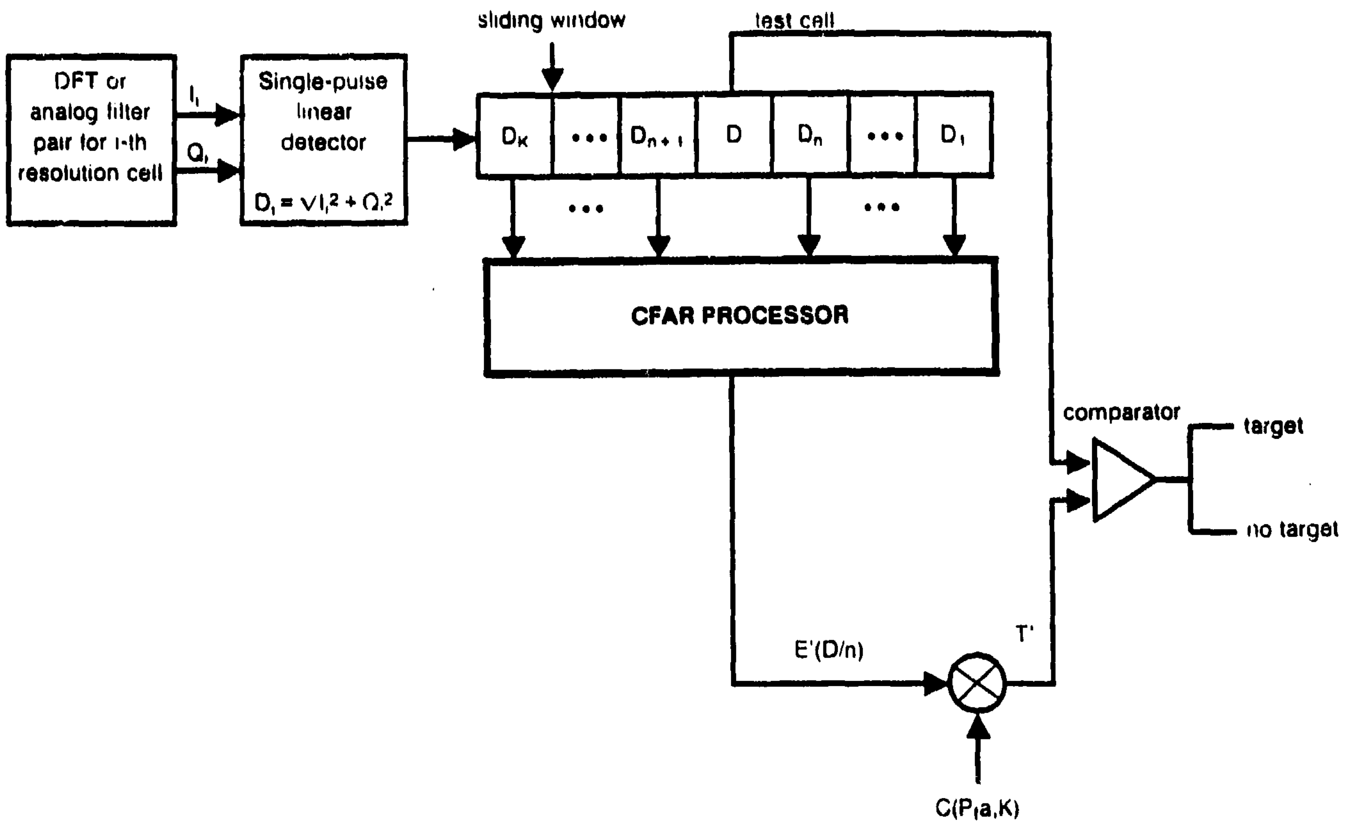


Figure 1. A typical CFAR processor.

used, although other pdf's (Weibull, log-normal) can be used to determine an appropriate threshold.

Cell Averaging (CA)

For CA CFAR, an average value over the sliding window is used to determine the background level for setting a decision threshold. The typical assumptions in CA are that the clutter is homogeneous over the sliding window and that the power values form a random sample [i.e., an independent, identically distributed (iid) collection of random variables]. Various distributions can be used to model the background clutter distribution, such as Rayleigh, Weibull, log-normal, or K-distributed. When the underlying assumptions are met, CA has the lowest CFAR loss. However, when the background clutter does not meet the assumptions, for example, due to inhomogeneities in the clutter power, the false alarm rate can be substantially higher than the desired P_f.

Censored CA (CCA)

Censored CA is a variation of the basic cell averaging (CA) CFAR, proposed by Rickard and Dillard (1974) to address the problem of masking by multiple targets. The envelope values D_i are ranked and the k largest values are censored from the sliding window. The normal cell averaging procedure is then followed on the censored sample. The excision CFAR procedure, described by Goldman and Bar-David (1988) is not significantly different than CCA is its processing or assumptions.

Censored Greatest Of (CGO)

The Greatest Of (GO) CFAR is described in later section. The CGO is a variation of GO, suggested by Weiss (1982). GO works well for clutter edges but suffers when interfering targets are present. In CGO, the k largest values are censored from each side of the split sliding window. As long as the number of interfering targets in each window is less than k, CGO works better than GO. However, additional losses are incurred due to the smaller sample size.

Clutter Map (CM)

The sliding window commonly used for CFAR consists of adjacent range and Doppler cells. Clutter map CFAR uses previous values at the same range cell for the sliding window. This is appropriate when the envelope data are inhomogeneous in range or in adjacent Doppler cells. To reduce memory requirements, the method proposed by Nitzberg (1986) uses a exponentially smoothed estimate from previous scans. CFAR loss decreases with longer memory in the exponential filter. Losses vary from about 0.3 dB for a weight of 0.98 to over 15 dB for a weight of 0.125.

Censored Mean Level Detector (CMLD)

The censored mean level detector (CMLD) is a variation of MLD that uses censoring to reduce the effects of a nonhomogeneous clutter/noise environment. In CMLD, the k largest noise samples are discarded prior to sample averaging.

Censored Video Integrator (CVI)

Video integration is a noncoherent averaging of radar signal amplitudes. In censored video integration, only the k smallest samples are included in the average. The first k-1 are given unit weight while the kth sample is weighted by N - k + 1 to give a unbiased, minimum variance estimate of the average. CVI was proposed as a compromise to the complication of ordered statistics. Censoring incurs additional CFAR losses, on the order of several dB. CVI performance degrades if the number of interfering targets is larger than the number of censored cells.

Doppler Domain Localized Generalized Likelihood Ratio (DDL-GLR)

Doppler domain localized generalized likelihood ratio (DDL-GLR) is a CFAR algorithm to efficiently detect moving targets in strong clutter with a complicated spectrum. DDL-GLR assumes that measurements are available to estimate unknown clutter statistics. DDL-GLR operates on the Fourier transform of the I and Q samples. For DDL-GLR, areas where the spectrum is rapidly changing are delineated [termed regions of detection improvement (RODIs)]. A GLR algorithm is then applied within these regions. DDL-GLR assumptions are identical to those for GLR, i.e., the data is divided into primary and secondary sets. The secondary data are assumed to have noise and clutter only (no targets) while the primary data set could have a target. The covariance matrix is estimated from the secondary data set. The noise plus clutter is assumed to have a complex Gaussian distribution.

Distribution Free (DF)

Distribution free detectors are usually equivalent to nonparametric detectors in the engineering literature (Kassam, 1980). The discussion of these methods is in the section on nonparametric methods.

Generalized Cell Averaging (GCA)

Generalized cell averaging (GCA) is a variation on the ubiquitous CA CFAR. A basic assumption under CA is that the data in the reference (sliding) window are independent and identically distributed. For the practical situations of weather clutter or chaff, the samples may be correlated. When the samples in the reference window are correlated, the normal CA procedure estimates a threshold that is too high. In the GCA study by Himonas, et al. (1990), the effects of spatial correlation of the samples in the reference is modeled as a first-order Markov process. The covariance matrix is estimated from the reference window. GCA can thus adapt to the actual correlation structure of the clutter.

Generalized Censored Mean Level Detector (GCMLD)

The presence of interfering targets in the reference windows can raise the threshold too high in many CFAR algorithms and result in masking of targets. GCMLD is a modification of CMLD that estimates the number of interfering targets and censors them from the reference sample. A Swerling 2 target in a white Gaussian background is assumed. A sequential censoring procedure is applied to the leading and lagging windows to remove the samples from the interfering targets.

Generalized Likelihood Ratio (GLR)

Generalized likelihood ratio (GLR) is an adaptive procedure for detecting targets. Two sets of input data are considered. First, the primary data set may contain a signal; the secondary data set is assumed to contain only noise or clutter. The covariance matrix for both data sets are assumed to be the same. A likelihood ratio is then formed of the maximum likelihood function under the hypothesis of no target to the maximum likelihood function under the hypothesis of a target. If the ratio exceeds some threshold, a target is declared. The primary and secondary data are assumed to iid complex Gaussian variates. GLR was found by Cai and Wang (1991) to offer more robust performance than SMI when the background was colored Weibull or log-normally distributed.

Greatest Of (GO)

Greatest Of (GO) CFAR is a modification of CA to minimize effects of clutter edges. The cells in the reference window is split into leading and trailing sets. The average value is estimated for both sets. The parameter for determining the clutter distribution and the CFAR threshold is selected as the maximum of the two averages from the leading and trailing half-windows. GO thus provides some control of P_f in clutter edges with relatively small CFAR loss over that of CA. However, an interfering target in one of the half-windows decreases P_d intolerably and the target is masked.

Generalized Sign Test (GST)

The generalized sign test (GST) counts the number of times the value in the test cell exceeds the M values in the reference window during the N observation intervals in a CPI. The number of times the N test cells exceeded the M reference cells is compared to a threshold and a detection declared if it exceeds the threshold. The threshold value can be set to provide the required P_f. GST assumes the N test cell values and the NxM reference cell values are independent and identically distributed.

Generalized Two-Level Censored Mean Level Detector (GTL-CMLD)

GTL-CMLD is an extension of CMLD to model both nonhomogeneous clutter and interfering targets. The data from the reference cells are ranked by magnitude and then processed by a two-level censoring algorithm. For the first censoring level, a determination is made whether the test cell is in the noise or the clutter. The other type is censored from the reference cells. For the second level of censoring, the number of interfering targets is determined and censored from the reference cells. The remaining cells form the estimate of the clutter/noise level in the test cell.

Heterogeneous Clutter Estimating (HCE)

The heterogeneous clutter estimating CFAR is a modification of CA and assumes that a heterogeneous clutter field can be represented as two homogeneous clutter fields with different levels. The transition cell between the fields and the clutter levels in the two fields is estimated. Research on the use of HCE showed that inevitable errors in the location of the transition cell can place the test cell in the wrong clutter field and lead to false alarms or missed detections. A biased version, that favors placing the test cell in the more intense clutter field, appears to help.

Logarithmic (LOG)

Generally, LOG CFAR applies the cell averaging technique to samples from a logarithmic detector. The advantages of logarithmic detection are the larger dynamic range and normalization of data by subtraction rather than division. The disadvantage of LOG CFAR is that the CFAR loss is higher than linear for the same number of cells in the reference window.

Log-t Detection

The log-t detector maintains a constant false alarm rate when the clutter cells are distributed log-normally or Weibull. The clutter is assumed to be homogeneous (identically distributed) and independent. As with most CFAR algorithms, the reference window can include range and/or Doppler cells. A test statistic, based on the difference between the test cell and the average over the reference window in log space, is compared to a threshold. If the test statistic exceeds the threshold, a detection is declared. The log-t name is suggested because of the log transformation and the fact that it has a Student-t distribution when the clutter is log-normal. Goldstein (1971) shows that the log-t statistic is independent of the scale and shape parameters of the Weibull distribution.

Modified Greatest Of (MGO)

The modified greatest of (MGO) CFAR uses the standard GO technique but with an estimated value of the Weibull shape factor from the reference window.

Modified Generalized Sign Test (MGST)

MGST is a modification of GST to control the false alarm rate in correlated and nonhomogeneous clutter. The first part of the CFAR processor, as described by Trunk, et al., (1971), is similar to the GST, described above, but implemented in hardware. A two threshold technique is used to control false alarm rate. The first threshold is set for a P_f of 10^{-6} when the clutter data are actually independent and identically distributed. The second threshold is claimed to estimate the standard deviation of the cells in the reference window and to remove extraneous targets from the reference window. The summary of the Trunk article notes that preliminary results are encouraging but that no quantitative statements can be made yet.

Mean Level Detector (MLD)

The mean level detector provides an adaptive estimate of the background noise level by the sample mean of surrounding reference samples. MLD was originally implemented as a thresholding circuit for a short pulse noncoherent radar. MLD assumes a homogeneous reference channel, consisting of square-law detected Gaussian noise samples, all with the same variance as the test cell.

Modified Sample Matrix Inversion (MSMI)

In MSMI, the sample matrix inversion technique is modified to allow for colored Gaussian clutter. An analysis by Cai and Wang (1991) shows that MSMI and GLR offer the same performance when the clutter has a Gaussian distribution. MSMI is easier to implement than GLR. However, GLR has better performance when the clutter has a non-Gaussian distribution, such as Weibull or log-normal.

Mann-Whitney (MW) Detector (Two Sample Wilcoxon)

The Mann-Whitney detector is similar to GST and counts the number of times the value in the test cell exceeds values in the reference cells. However, the Mann-Whitney detector compares a given test cell to all of the reference cells in the CPI. The total count is compared to a threshold and a detection declared if it exceeds the threshold. The threshold value can be set to provide the required P_f. The Mann-Whitney detector assumes the M test cell values and the NxM reference cell values are independent and identically distributed. The Mann-Whitney detector has better performance than GST but with a more complex design.

Maximum Family (MX) of CFAR Detectors

Several types of CFAR detectors that use a maximum of 2 local clutter level estimates have been considered by Ritcey, et al. (1991) for false alarm control at clutter edges. MX mean level detection (MX-MLD) is equivalent to GO. In MX censored mean level detection (MX-CMLD), winsorized (Ritcey, op.cit, Eq. 5) mean estimates are used for the 2 reference half-windows. For MX ordered statistic detection (MX-OSD), the maximum of the quartiles for the two half-windows is used to treat interfering targets. Collectively, these detectors are referred to as MX linear combination detectors (MX-LCD). The local estimates for the two half-windows are weighted linear combinations of the data in the reference cells. The reference cell samples are assumed to be iid. A simulation comparison shows that MX-CMLD and MX-OSD are similar in performance. MX-OSD is more complicated because of the need for sorting and has slightly higher loss.

Nonparametric Detectors (NP)

Nonparametric CFAR techniques are used to provide a constant false alarm rate when the clutter or background noise statistics are unknown. Nonparametric and distribution-free techniques are taken to be equivalent in the CFAR literature, although this is not strictly true. As with most CFARs, NP methods involve computing a detection statistic from the reference cells, which is then compared to a threshold. Typically, NP methods make only weak assumptions about the clutter, such as that the samples are independent, identically distributed with known median. Two NP methods are the sign test and the Wilcoxon detector. These can't be applied to radar detection because of practical considerations (i.e., the median level and the phase are unknown). For CFAR, the generalized sign test and the Mann-Whitney detector are used. These CFARs are described in their separate sections.

Ordered Statistic (OS)

Ordered statistic (OS) CFAR uses ordered statistics to improve performance with clutter edges and interfering targets. The data in the reference cells are ranked and the kth largest sample is used for forming the detection threshold. For large values of k, OS can control the false alarm rate similarly to GO but with lower detection efficiency. The performance of OS with interfering targets is similar to SO. For determining the threshold, the samples from the reference cells are assumed to be independent, identically distributed. Various distributions have been considered, such as exponential (for square law detection), Rayleigh, and Weibull. Typical CFAR losses as compared to CA are small, about 0.5 dB.

Polarimetric

The typical CFAR algorithm processes the amplitude of the radar return. Additional information is contained in the complete scattering matrix for the target or clutter. Since measurements of the scattering matrix are not available for the AI-CFAR program, this method is not considered.

Polynomial

In some instances, the clutter power can vary tremendously over a few kilometers. In polynomial CFAR, it is assumed that the logarithm of the clutter power varies as a low order polynomial. CFAR losses can be 10 to 20 dB if a small number of samples (less than about 16) are used for fitting the polynomial. CFAR losses also increase as the degree of the fitted polynomial increases.

Range-Azimuth

A study by Martin (1976) considered the use of reference cells in both range and azimuth for a CA CFAR algorithm. An experiment measured the improvement in detection performance for S and L band land-based radar. A typical improvement in detection was about 5 dB.

Sample Matrix Inversion (SMI)

The sample matrix inversion CFAR is similar in operation to the GLR. The input data are divided into the primary and secondary data sets. The primary may contain a target while the secondary is assumed to composed of clutter and noise only. The covariance matrix of the clutter/noise is estimated from the secondary data set. A weight vector is computed from the secondary data and the inverse of the sample covariance matrix. A test statistic is formed from the weight vector and the primary data. When the test statistic exceeds some threshold, a detection is declared. SMI assumes that the primary and secondary data are independent, identically distributed complex Gaussian random variables with the same covariance matrix.

Smallest Of (SO)

Smallest Of (SO) CFAR is a modification of CA to minimize the effects of interfering targets. The cells in the reference window are split into leading and trailing sets. The average value is estimated for both sets. The parameter for determining the clutter distribution and the CFAR threshold is selected as the minimum of the two averages from the leading and trailing half-windows. SO thus provides better performance over CA with interfering targets in a homogeneous background. However, SO shows severe degradation in the presence of a clutter edge.

Scan by Scan Averaging (SSA)

Scan by scan averaging (SSA) CFAR uses reference cells in a homogeneous clutter area, called a map cell. Data from several scans are used, so an assumption of time invariant statistics on a scale of several seconds is implied. With the additional data from previous scans, the spatial window can be much smaller for the same CFAR loss, so that the assumption of homogeneous clutter is more likely to hold. The normal CA algorithm is used on the data in this expanded reference window. However, targets that remain in the same range cell are self-masking and present a problem for SSA. SSA does show some improved performance over CA in nonhomogeneous clutter environments.

Trimmed Mean (TM)

Trimmed mean CFAR is a generalization of OS. The samples in the reference window are ordered as in OS. The smallest and largest values are censored and the remaining cells averaged together. TM performance is thus comparable to OS for clutter edges and for interfering targets. TM has a very slight improvement over OS for a single target in homogeneous background.

Weber-Haykin (WH)

The Weber-Haykin CFAR algorithm is an extension of OS CFAR for two parameter distributions. The samples from the reference window are assumed to be independent, identically distributed. The power and skewness parameters of the Weibull distribution are estimated from ranked samples and a threshold determined. The CFAR loss for WH is larger than that for a single parameter OS, where the skewness is known a priori. The single parameter CFAR (OS) should be used when the skewness is close to some a priori value and the number of sample is small. The CFAR loss is also affected by the rank of the samples used for parameter estimation. The lowest CFAR loss occurs when the smallest and largest ranks are used; however, if these are used the robustness to clutter edges and interfering targets will be lost.

Table 3. CFAR algorithm articles.

AUTHOR	YEAR	CFAR ALGORITHMS	SITUATIONS	ASSUMPTIONS	PERFORMANCE MEASURES	RECOMMENDATIONS/NOTES
Al-Hussani, E.K	1988a	SO, GO	• M pulses coherently integrated • Sharp clutter edges • Interfering targets	• Gaussian noise • Swerling target	P_d vs SNR	• Comparison of SO and GO • For large sliding windows (N~10) and homogeneous environment performance of SO and GO are similar
Al-Hussani, E.K	1988b	GO, CGO	• Interfering targets	• Gaussian noise • Swerling I Target	P_d vs SNR, INR	• CGO - good performance if number of censored cells ≥ number of interfering targets
Al-Hussani, E.K	1988c	OS	• Target in sea clutter	• Log normal clutter • M integrated pulses • N range cells	CFAR loss table	Losses large (> 10 dB) for small N and large α (shape parameter)
Bakulev, P A et al	1989	CA, SO, GO, OS	• Model 1-Homogeneous and stationary • Model 2-Inhomogeneous • Model 3-Homogeneous with interfering targets	Standard CFAR assumptions	CFAR loss	• No single processor works best for all types of noise considered • Maps needed of clutter types in the observation space • CA good for model 1 • GO shows improvement for models 2 and 3 but clutter edges and interfering targets are a problem • OS reduces effects of masking by interfering targets • Promising direction for processor design is adaptive robust algorithms for inhomogeneous non-Gaussian noise
Barkat, M. and Varshney, P K	1991	CA	Multiple sensors	• Gaussian noise • Swerling I target	P_d vs. SNR	Improved performance with additional detectors
Barkat, M and Varshney, P K	1988	CA, SO, GO multiple background estimators	• Unknown noise level • Frequency diversity	• Gaussian noise, unknown level iid • Swerling I target	P_d vs. SNR	Increased P_d with multiple background estimators
Blake, S	1988	OS	• Interfering targets • Clutter edge	• Nonuniform Rayleigh clutter • Multiple Swerling I targets	CFAR loss for ideal and for interfering targets	• Interfering target - single quartile • Clutter edge - single quartile
Bucciarelli, T	1987	GO, CA, MGO	Weibull clutter with various shape factors	• Weibull clutter • Shape factor estimation for modified	• CFAR loss	• GO works well with Weibull clutter • MGO losses high for Rayleigh clutter
Cai, L and Wang H.	1991	SMI, MSMI GLR	Colored interference with unknown statistics	Complex Gaussian noise	P_d vs. SINR, spectral spread, number of pulses	• MSMI and GLR equivalent for Gaussian interference • GLR is more robust for colored Weibull and log-normal interference
Conte, E.M. Longo, and Lops, M.	1989	CCA	Multiple targets	Rayleigh clutter	P_d vs. SNR	CCA better than SO for multiple interfering targets
Conte, E.M Longo, and Lops, M	1988	CA	K-distributed clutter	• Swerling targets • Compound Gaussian clutter	• CFAR loss • P_d vs SCR	• Compound Gaussian clutter common for high resolution radar • CA CFAR loss high for small N
Dillard, G.M.	1974	MLD Binary integration	Swerling 0	• Narrowband Gaussian noise • Nonfluctuating target	• Single pulse P_d vs. SNR • P_d vs. SNR for M out of N detection	CFAR loss decreases with - decreasing P_f - increasing N
Dillard, G.M. and Antoniak, E	1970	Distribution free	Swerling 0,2,4 targets	iid random variables in multiple range bins	P_d vs SNR	Recommended for some jamming situations
Elias-Fuste, A R and deMercado, M G and de los Reyes Davo, E	1990	OS, GO	• Uniform clutter • Clutter edges • Multiple targets	Exponential clutter, iid	P_d vs. SNR	GO works well in nonhomogeneous and multiple target situations

Table 3. CFAR algorithm articles.

AUTHOR	YEAR	CFAR ALGORITHMS	SITUATIONS	ASSUMPTIONS	PERFORMANCE MEASURES	RECOMMENDATIONS/NOTES
Finn, M., et al	1986	HCE	Two clutter fields of different powers in sliding window	Rayleigh, log normal clutter	P_d vs. SNR	• Estimate of transition cell introduces additional error considerations • HCE perform better than CA in heterogeneous clutter • Small loss in uniform clutter
Finn, M., et al.	1968	CA	Targets in extended clutter	• Rayleigh clutter iid RVs • Swerling 0,1,3	P_d vs. SNR for various N (N = number of cells in sliding window)	Analysis for edge effects, two targets
Gandhi, P.O., et al.	1988	CA, GO, SO, TM, OS	• Clutter edge • Multiple targets	Exponential clutter iid RVs	P_d vs. SNR	OS-CFAR superior for multiple targets
Goldman, H	1990	Excision CFAR	Interfering targets	• Exponential clutter • Swerling 0, 1 targets	P_d vs. INR	Excision CFAR superior to SO for interfering targets
Goldman, H., et al	1988	Excision CFAR	Interfering targets	Exponential clutter	• P_d vs. SNR • P_d versus P_f	Excision CFAR reduces effects of interference
Goldstein, G B	1973	log-t test	log normal, Weibull clutter	• Homogeneous clutter • Rayleigh target	CFAR loss	• log-t maintains CFAR in log-normal and Weibull • log-t can, under certain conditions, be used for the entire family of log-normal and Weibull
Hansen, V G et al	1980	GO	Homogeneous clutter	• Swerling 0, 1, target • P_d versus SNR	CFAR loss	CFAR loss for GO is typically 0 1 to 0 3 dB
Hansen, V G et al	1972	LOG		Stationary Gaussian noise	P_d vs SNR	• Advantages - dynamic range - normalization by subtraction • Disadvantage - poorer detectability for fixed sample size
Hansen, V.G. et al	1971	• 2 sample sign test • 2 sample Wilcoxon	Stationary Gaussian noise	iid RVs	P_d vs SNR	Mann Whitney better than sign test, but much more complicated
Harkness, L.L. et al	1989	CA, SO, GO, OS	Nonhomogeneous interference	Standard CFAR assumptions	SNR vs. range bin for various scenarios	• CA - minimum loss for homogeneous clutter • GO - good for clutter edges but target masking by strong interference • SO - large CFAR loss and too many false alarms at clutter edges • OS - larger CFAR losses than CA (typically 0 5 dB) - no masking effects if k and N appropriately chosen - robust to multiple target and jet engine modulation [illegible] • Classified discussions of ECM
Hendon, E.J. et al	1990	SLC (side lobe canceler)		Interference is non-stationary but slowly varying Gaussian random process	P_d vs. SNR	
Himonas, S.D. et al	1990	CA, GCA		• Clutter is partially correlated Rayleigh • Noise is uncorrelated Rayleigh	P_d vs. SNR for various correlation coefficients	GCA works well for correlated clutter

Table 3. CFAR algorithm articles.

AUTHOR	YEAR	CFAR ALGORITHMS	SITUATIONS	ASSUMPTIONS	PERFORMANCE MEASURES	RECOMMENDATIONS/NOTES
Himonas, S D et al.	1990	GTL-CMLD	Time diversity (several pulse per) Space diversity (several receivers)	• Rayleigh targets (could be correlated) • Clutter power may have transitions	P_d vs SNR from simulations	M out of L fusion rule - better than AND, OR fusion rules - 3 out of 4 rule appears less sensitive to correlation in target returns
Himonas, S D et al	1989	GCMLD	Interfering targets	• Swerling II targets • White Gaussian noise	P_d vs SNR for various sliding window lengths	• More robust for both strong and weak interfering targets • Robust even for a small number of cells in the sliding window
Kelly, E J	1986	GLR	Unknown Gaussian noise	Complex Gaussian noise	• P_d vs SNR (*parametric curves*) • CFAR loss vs sample size	CFAR logs decreases with increasing sample size
Lefferts, R E	1981	Double Threshold	Nonstationary and azimuthally correlated clutter	Output of first detector is Markov process	P_d vs. P_f	Empirical optimization required for the second threshold
Lei, S., et al.	1989	OS, OSGJ, OSTWO, OSTA	Clutter edges, interfering targets	Log-normal clutter	P_d vs. SNR	None.
Levanon, N., et al	1990	OS, WH	Weibull clutter		P_d vs. SNR	
Levanon, N.	1990	CVI	Interference (discrete)	• Swerling II target • Gaussian noise	$\ln \frac{P_d}{1-P_d}$ vs. SNR	CFI better than binary integration - perform well for interfering clutter - smaller detection loss
Levanon, N., et al.	1989	OS, WH	Weibull clutter	Weibull distribution - known shape parameter both parameter unknown	P_d vs. SNR	• Estimation of both parameters of Weibull distribution is very costly in terms of CFAR loss • Use OS-CFAR when interfering targets are present (or suspected)
Lops, M., et al	1989	SSA	Nonhomogeneous clutter	• Clutter in a given map cell is time invariant • Swerling II target • Independent Gaussian clutter (Rayleigh developed)	• P_d vs. SNR • P_d vs. number of scans with SNR as a parameter	Self masking targets (i.e., those which remain in one range cell) are a problem.
Martin, G R	1976	Adaptive threshold		• Recorded clutter • Constant amplitude target with random phase		Range and azimuth method better than range only
Nitzberg, R	1986	CM	Nonhomogeneous clutter (in range/Doppler)	• Gaussian clutter • Clutter map is exponential weighting of prior scan measurements	• P_d vs. SIR • CFAR loss vs weighting parameter	CFAR loss depends on exponential weight
Nitzberg, R	1973	Polynomial environment	Locally nonstationary clutter	Polynomial variation of log (clutter power)	• CFAR loss vs. number of samples • P_d vs. SIR	Loss increases with degree of polynominal and decreases with number of data.
Nitzberg, R.	1972	MLD geometric mean	Unknown levels, unknown shape, geometric symmetry, geometric mean	• Rayleigh interference • Swerling I target	• P_d vs. SIR, SNR	CFAR algorithms for • unknown level • unknown slope • geometric symmetry

Table 3. CFAR algorithm articles.

AUTHOR	YEAR	CFAR ALGORITHMS	SITUATIONS	ASSUMPTIONS	PERFORMANCE MEASURES	RECOMMENDATIONS/NOTES
Novak, L.M	1980	Log CFAR, area CFAR	Strong interference	• Swerling II target • Homogeneous Rayleigh	• P_d vs. SCR • CFAR loss vs ≠ samples	Log works better when other targets, bright clutter cells, decoy reflectors, etc., are present
Rickard, J T., et al	1977	MLD, CCA	Multiple targets	Swerling 0, 2, 4 targets	P_d vs. SNR	Censoring for eliminating interfering targets
Ritcey, J A., et al	1991	MLD, MX-LCD, MX-CMLD, MX-OSD	• Clutter fields • Multiple targets	• Swerling II targets • Exponential clutter	CFAR loss vs number of reference cells, censoring points	OSD and CMLD performance is similar for typical multiple target, clutter patch simulations
Ritcey, J.A	1990	MX-MLD	Clutter edges	• Swerling 0, 2 targets • iid Gaussian noise	CFAR loss vs. number of reference cells	Exact probabilities calculated for MX-MLD including noncoherent integration
Rohling, H	1983	OS	• Clutter edges • Multiple targets	• Swerling I, III • iid Rayleigh clutter	SNR vs. range curves for CA, GO, OS	OS overcomes problems due to clutter inhomogeneity and multiple targets
Saniie, J et al	1990	OS		• Chi square target in Weibull clutter • Rayleigh target in Rayleigh clutter	• SNR for P_d = 0.5, P_f = 10^{-3} vs ≠ • samples for • n-pulse, OS filters	• Selection of OS versus n-pulse integrator depends on distributions of target and clutter • Similar distributions - n-pulse • Dissimilar distributions - OS
Sekine, M. et al	1989	CA	Sea clutter	Weibull clutter	None	None
Shore, M et al	1991	OS, CA, CCA	Multiple targets	• Swerling II target • Weibull clutter	• ln $[P_d/(1-P_d)]$ vs. SNR • CFAR loss vs. rank	OS performance similar to CMLD
Steenson, B O	1968	MLD	Nonstationary clutter	Swerling II target	P_d vs. SNR	
Tantaratana, S	1990		Assumes a symmetric probability density function for the background noise			
Trunk, G V., et al	1989					Consistency checks applied to amplitude and Doppler
Trunk, G V	1983	• Adaptive thresholding • Nonparametric detection • CA • Binary integrator • Clutter map	Review of current techniques	iid random sample	Qualitative	• CA - noise is Rayleigh • Binary integration - doubt about noise being Rayleigh distributed • ratio detector - Pulse to pulse noise variations • Clutter Map - better interclutter visibility for land based radars • Colored or non-Rayleigh noise - Two parameter estimations and more complicated adaptive detection
Trunk, G V	1978	CA, CCA	Multiple targets, closely spaced		Monte Carlo detection counts	
Trunk, G V., et al.	1974	GST, MGST	• Nonhomogeneous clutter • Correlation between pulses (weather clutter or MTI output)	• iid samples in reference cells • Swerling I and III targets	P_d vs. SNR with extraneous targets as a parameter	Quantitative evaluation not available
Trunk, G.V., et al.	1970	CA	High resolution radar	• Log normal clutter • Contaminated normal	P_d vs. SNR	Use median detector and horizontal polarization for high resolution radar.

Table 3. CFAR algorithm articles.

AUTHOR	YEAR	CFAR ALGORITHMS	SITUATIONS	ASSUMPTIONS	PERFORMANCE MEASURES	RECOMMENDATIONS/NOTES
Wang, H., et al	1991	DDL-GLR	Severely nonhomogeneous environments	Gaussian interference	P_d vs. SINR	Further work.
Wang, H., et al	1991	GLR, SB single band MB multiband	• Nonhomogeneous environments • Interference	Complex Gaussian signal and interference	P_d vs. SINR (signal to interference plus noise ratio)	Further investigation needed on application of MBGLR to airborne surveillance radar.
Wang, H., et al	1990	SMI	Nonstationary/non-homogeneous clutter	Complex Gaussian target and clutter	P_d vs. SINR	Multiband better than single band
Wang, H., et al.	1990	DDL-GLR	• Severely nonhomogeneous clutter • Small data samples	Gaussian interference	P_d vs. range bin	Further work
Wanielik, G. et al.	1990a		Multi-target Nonhomogeneous		None	Inconclusive.
Wanielik, G. et al.	1990b	P		Complex normal for target and clutter	None	
Wanielik, G. et al	1990c	P		Complex normal for target and clutter	None	
Weber, P. et al.	1985	OS-two parameter	• Nonhomogeneous clutter • Multiple targets	Weibull clutter	CFAR loss	For small samples and small variation in skewness, use single-parameter OS.
Weiss, M	1982	CA, GO, SO	• Clutter edges • Multiple targets	iid Gaussian noise Rayleigh targets	P_d vs. SNR	• Further work on CFAR supersedes the conclusions • Note: larger M reduces CFAR loss but a nonhomogeneous sliding window is more likely